VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroj pro klonování a mazání databázových dat

Database Data Cloning and Deleting Tool

2010 Lukáš Adámek

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Karviné_____ Podpis _____

Poděkování

Chtěl bych tímto poděkovat panu Ing. Filipu Perglerovi za vedení mé bakalářské práce a panu Ing. Janu Šottovi, pod jehož vedení jsem práci začínal. Dále patří poděkování všem kolegům, kteří se podíleli na testování aplikace.

Abstrakt

Cílem této bakalářské práce je vyvinout vhodné algoritmy pro klonování a mazání dat. Na jejich základě pak bude vytvořena desktopová aplikace umožňující snadné klonování a mazání dat v databázích Microsoft SQL od verze 2005. Obě operace musí podporovat omezující podmínky, tj. podmínky konkretizující klonovaná či mazaná data. Nutná je také podpora ukládání konfigurací. Po dokončení bude aplikace primárně sloužit pro testování informačních systémů vyvíjených ve společnosti T-SOFT a.s. Je třeba brát v úvahu, že aplikace může pracovat s většími objemy dat (statisíce až miliony záznamů).

Abstract

Main goal of this work is development of data cloning and data deleting algorithms. These algorithms will be used in desktop application for easy cloning and deleting data in Microsoft SQL since version 2005. Both operations must allow specification of data concretization conditions. Configurations saving must aslo be possible. Primary use of application will be testing of information systems developing in T-SOFT a.s. company. Application should be useable for relatively large amounts of data (hundreds of thousands to millions of records).

Klíčová slova

databáze, kopie, klon, klonování, mazání, desktop, vývoj, vývojový nástroj, informační systém

Keywords

database, copy, clone, cloning, deleting, desktop, development, development tool, information system

Obsah

1 Úvod

Databáze dnes tvoří základ mnoha informačních systémů. Vývoj každého takového systému přináší nezbytné testování, které probíhá nejčastěji na interních serverech firmy. Testovací data si vývojáři zadávají do systému sami nebo si je vygenerují. V případě pozdějších fází vývoje, kdy už je určitá verze systému v provozu, mohou rovněž od zákazníka obdržet část ostrých dat.

Nad takto připravenou databází poté probíhají různé testy, ať už funkčnosti či výkonu. Převážně u výkonnostních testů je žádoucí, aby každé jejich opakování probíhalo na stejných výchozích datech. Toho však není snadné dosáhnout, protože např. test hromadného importu dat vloží data do mnoha tabulek. Uvádění databáze do původního stavu pak nezřídka představuje zdlouhavou práci. Řešení pomocí provedení zálohy databáze před prvním spuštěním testu a jejím obnovení před každým dalším spuštěním je nepohodlné a ne vždy možné. Stejně tak vymazání obsahu databáze a nové ruční vložení či vygenerování dat.

Usnadnění by mohl přinést nástroj pro klonování a mazání dat. Vývojář by si za účelem testování mohl tímto nástrojem vytvořit kopii potřebných dat (ať už vytvořených jím samým nebo zadaným do databáze jakýmkoliv jiným způsobem). Po provedení potřebných testů pak tuto testovací kopii může odstranit a případně si z originálních dat vytvořit opět kopii novou. Tento způsob testování, kdy nedochází k ovlivňování originálních dat, může být výhodně využíván také pří práci více vývojářů na jednom informačním systému. Každý vývojář se může zabývat testováním jím vytvářené funkcionality bez obav, že ovlivní data, která potřebuje kolega pracující na jiné funkcionalitě.

Dalším možným využitím aplikace pak může být promazání objemné databáze, například za účelem vytvoření odlehčené, prostorově nenáročné, verze pro prezentace, demonstrační instalace apod. V omezené míře je možné použití i při specifických požadavcích zákazníků na promazání dat v jejich provozním prostředí.

V první části práce se pokusím přiblížit čtenářům problematiku klonování a mazání dat z pohledu uživatele. Ve druhé části se budu zabývat vyvinutými algoritmy pro klonování a mazání dat. Podíváme se i na omezení, která tato algoritmy mají. Třetí část je věnovaná stručnému popisu vlastního nástroje. Ve čtvrté – poslední – části uvedu některé poznatky z vývoje a testování a budu se zabývat výhledem na možný budoucí rozvoj aplikace.

2 Problematika klonování a mazání dat

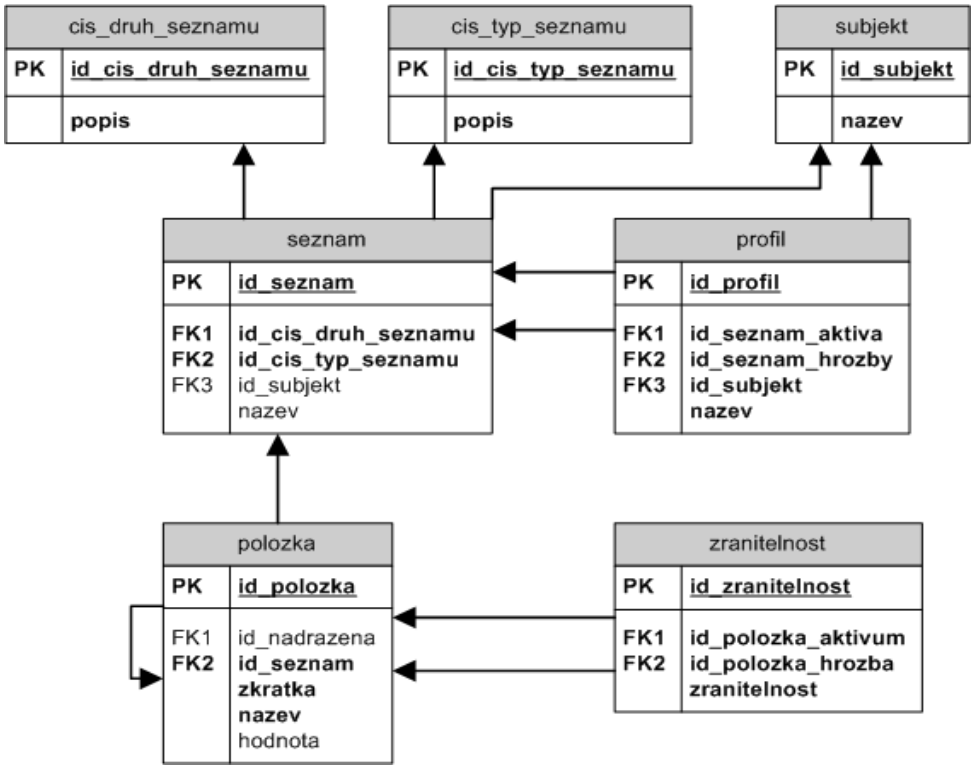
V této části práce se pokusím čtenáře uvést do problematiky klonování a testování dat. Zavedeme a definujeme si potřebné pojmy. Půjdeme cestou od hrubých definic k detailům. Ze všeho nejdříve se ale seznámíme se s ilustračními databázemi, které poslouží ke znázornění různých pojmů.

2.1 Ilustrační databáze

2.1.1 Riskan

K ilustraci některých uvedených poznatků využiji upravenou a zjednodušenou databázi firmou vyvinutého rizikového kalkulátoru Riskan. Pro snadnější pochopení si dovolím uvést stručný popis kalkulace rizik a samotné databáze.

Kalkulace rizik spočívá ve vytvoření profilu tvořeného aktivy (aktivum = to, co pro mě má nějakou hodnotu; například firemní server) a hrozbami (hrozba = to, co ohrožuje aktiva; například výpadek proudu nebo požár). Aktivům uživatel přiřadí jejich hodnotu, hrozbám jejich pravděpodobnost. Aktivum je ohrožováno hrozbou určitou mírou zranitelnosti. Kombinace hodnoty aktiva, pravděpodobnosti hrozby a zranitelnosti aktiva hrozbou udává výsledné riziko.



Obrázek 2.1

Tabulka *subjekt* eviduje subjekty – firmy – využívající kalkulátor rizik.

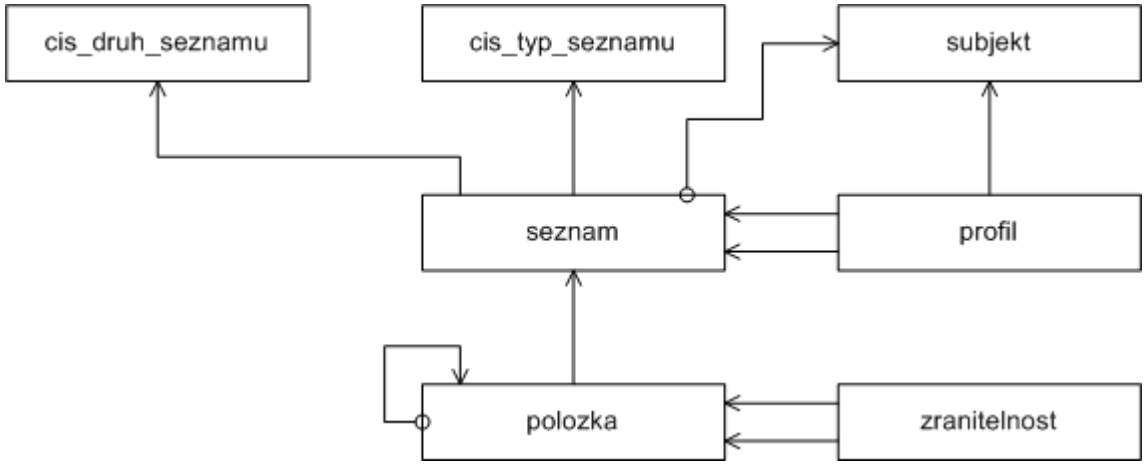
Tabulka *seznam* obsahuje pojmenované seznamy. Druh seznamu – šablona, vlastní nebo použitý – je určen vazbou na číselník *cis_druh_seznamu*. Typ seznamu – aktiva nebo hrozby – je určen vazbou na číselník *cis_typ_seznamu*. Použité seznamy a šablony nejsou vázány na subjekt, naopak vlastní seznamy patří jednomu konkrétnímu subjektu.

Tabulka *profil* obsahuje údaje o profilech rizik. Profil patří konkrétnímu subjektu a je tvořen seznamem aktiv a seznamem hrozeb. Při zakládání profilu v aplikaci uživatel (patřící do některého ze subjektů) vybírá aktiva a hrozby z vlastních seznamů a ze šablon. Po založení profilu je automaticky vytvořena kopie vybraného seznamu (včetně položek). Druh zkopírovaného seznamu je změněn na „použitý seznam“.

Tabulka *polozka* eviduje hierarchickou strukturu položek seznamu. Pokud položka patří do šablony nebo vlastního seznamu, je její hodnota vždy NULL. Pokud položka patří do použitého seznamu, je její hodnota NULL (uživatel zatím nevyplnil) nebo číslo, označující u položek seznamu aktiv hodnotu aktiva, u položek seznamu hrozeb pak pravděpodobnost hrozby.

Tabulka *zranitelnost* obsahuje záznamy o zranitelnosti aktiva hrozbou.

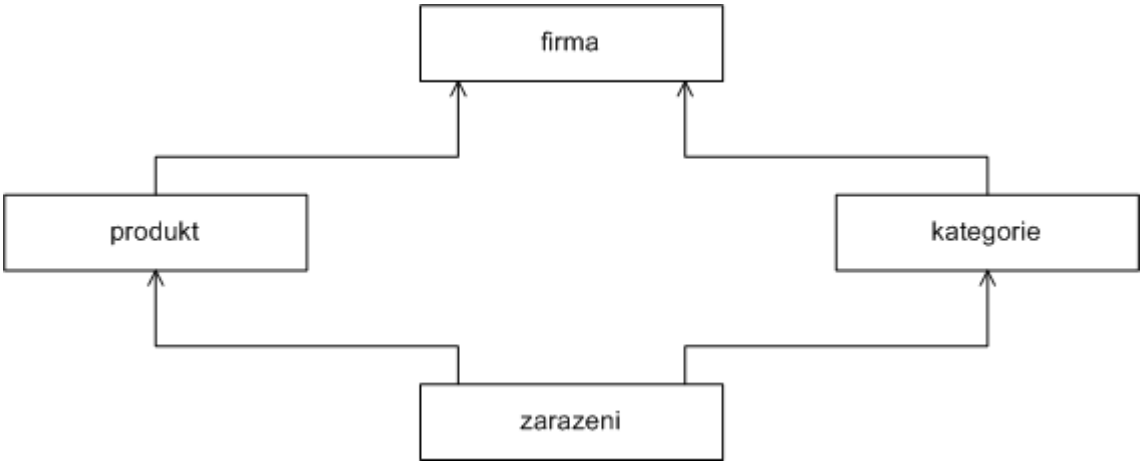
K ilustračním účelům budu používat jednodušší diagram. Šipka směřuje k tabulce primárního klíče. Prázdné kolečko značí, že záznam nemusí být členem vazby (jinými slovy – cizí klíč umožňuje hodnotu NULL).



Obrázek 2.2

2.1.2 Evidence firem

Dále se v textu setkáte s fiktivním schématem evidence firem a jejich produktů, které je možno řadit do kategorií (praktické využití dané podoby ponechme stranou).



Obrázek 2.3

2.1.3 Obecná fiktivní databáze

Do třetice pak některé poznatky ilustruji pomocí obecných fiktivních databází (tabulky A, B, C… bez specifikace sloupců a bez dat), vytvořených „na míru“ konkrétní ilustraci.

2.2 Základní popis operací

2.2.1 Klonování

Klonování je vytvoření kopie jednoho nebo více záznamů vybrané tabulky a všech záznamů, které jsou na tyto záznamy relačně vázány,

V praxi to znamená (např. pro evidenci firem), že klonování jedné konkrétní firmy je vytvoření kopie záznamu o firmě, vytvoření kopie záznamů o výrobcích a kategoriích této firmy a vytvoření kopie příslušných záznamů o zařazení výrobků do kategorií.

To, kterou firmu budu klonovat, specifikuji pomocí omezujících podmínek. Omezující podmínka je SQL klauzule where. Pro tabulku firem například `nazev like '%nábytek%'`. Samozřejmě můžu např. definovat ještě omezující podmínku na výrobky – `nazev like '%skříň%'` atd. Touto konkrétní kombinací naklonuji všechny firmy, které mají v názvu „nábytek“, z jejich výrobků pak klonuji pouze ty, které mají v názvu „skříň“.

Pro úplnost dodejme, že kopie záznamu není úplnou kopií v pravém slova smyslu, tj. kopie záznamu nemá hodnoty všech atributů shodné s originálním záznamem. Lišit se musí primární klíč i hodnoty cizích klíčů odkazujících na klonované záznamy.

2.2.2 Mazání

Mazání je odstranění jednoho nebo více záznamů vybrané tabulky. Tomu musí předcházet odstranění všech relačně vázaných záznamů.

V praxi (opět pro evidenci firem) odstranění firmy znamená odstranění všech záznamů o zařazeních, které se vážou na firmu přes tabulky kategorií a produktů, dále odstranění všech kategorií a produktů firmy a nakonec odstranění samotné firmy.

Zvláštní pozornost zde věnujme omezujícím podmínkám – pokud zadám omezující podmínku na tabulku produktů, nemusí jí vyhovovat všechny produkty odstraňované firmy. To však v konečném důsledku znamená, že záznam o firmě samotné smazán nebude.

2.3 Definice operací, zavedení a definice pojmů

Klonováním rozumíme vytvoření kopie všech operačních záznamů.

Mazáním rozumíme odstranění všech odstranitelných operačních záznamů.

Odstranitelný operační záznam je operační záznam, který lze operací mazání odstranit, tj. takový záznam, na nějž nejsou vázány žádné neoperační ani neodstranitelné operační záznamy.

Operační záznam je operační kandidát, nad kterým lze provést vybranou operaci. Operaci lze provést, pokud operační kandidát je operačním kandidátem všech operačních vazeb k rodičům s výjimkou vazeb majících v cizím klíči hodnotu NULL.

Operační kandidát je záznam, který je do operace zahrnut libovolnou operační vazbou. V případě běžných tabulek se mezi kandidáty dostanou pouze ty záznamy, které vyhovují omezující podmínce. U rekurzivních tabulek se v závislosti na nastavení rekurzivního zpracování tabulky mohou mezi kandidáty dostat i záznamy podmínce nevyhovující, protože omezující podmínka není řešena hned při detekci kandidátů, ale až později (o tom více v kapitole 2.4).

Omezující podmínka tabulky je SQL klauzule where, která může dále zužovat množinu operačních kandidátů (nikdy nemůže vést k jejímu rozšíření).

Operační vazba je vazba mezi operačními tabulkami. O cizím klíči takové vazby hovoříme jako o operačním cizím klíči.

Operační tabulka je tabulka spadající do operační oblasti.

Hlavní tabulka je tabulka určující základní operační oblast. Samotná operace klonování či mazání pak začíná od hlavní tabulky a pokračuje rekurzivně k potomkům nebo pasivním rodičům (více o klonovacím algoritmu v kapitole 3.3, více o mazacím algoritmu v kapitole 3.5).

Operační oblast je množina všech tabulek a vazeb mezi nimi, které budou zpracovány při provádění operace. Základní operační oblast tvoří hlavní tabulka a všichni její přímí a nepřímí potomci. Operační oblast je podmnožinou plně detekované oblasti. Uživatel může operační oblast upravovat vyjímáním a vkládáním tabulek a vazeb.

Plně detekovaná oblast je podmnožinou detekované oblasti. Obsahuje plně detekované tabulky a vazby. Je rozšiřitelná o částečně detekované tabulky a vazby, tedy o (v ní dosud neobsažené) přímé rodiče a přímé potomky v ní již obsažených tabulek. Rozšíření je umožněno díky tzv. aktivním vazbám.

Detekovaná oblast obsahuje všechny částečně i plně detekované tabulky a vazby. Dělí se na dvě podoblasti:

- **Podoblast 0** obsahuje všechny přímé i nepřímé potomky hlavní tabulky (tedy celou základní operační oblast), je pevně daná a nelze ji dále rozšiřovat.
- **Podoblast 1** obsahuje vše, co nepatří do podoblasti 0, tedy vše, co není potomkem hlavní tabulky. Vyjma stavu po prvotní inicializaci není pevně daná, lze jí rozšiřovat.

Plně detekovaná tabulka je tabulka, o níž známe všechny potřebné informace k jejímu zařazení do operační oblasti. Jedná se o sloupce a jejich vlastnosti, indexy a všechny vazby.

Částečně detekovaná tabulka je tabulka, o níž známe pouze základní informace potřebné k jejímu případnému zařazení do plně detekované oblasti (název tabulky a seznam sloupců, které jsou součástí cizích klíčů vázaných na tabulky z plně detekované oblasti).

Plně detekovaná vazba váže dvě plně detekované tabulky.

Částečně detekovaná vazba váže plně detekovanou a částečně detekovanou tabulku.

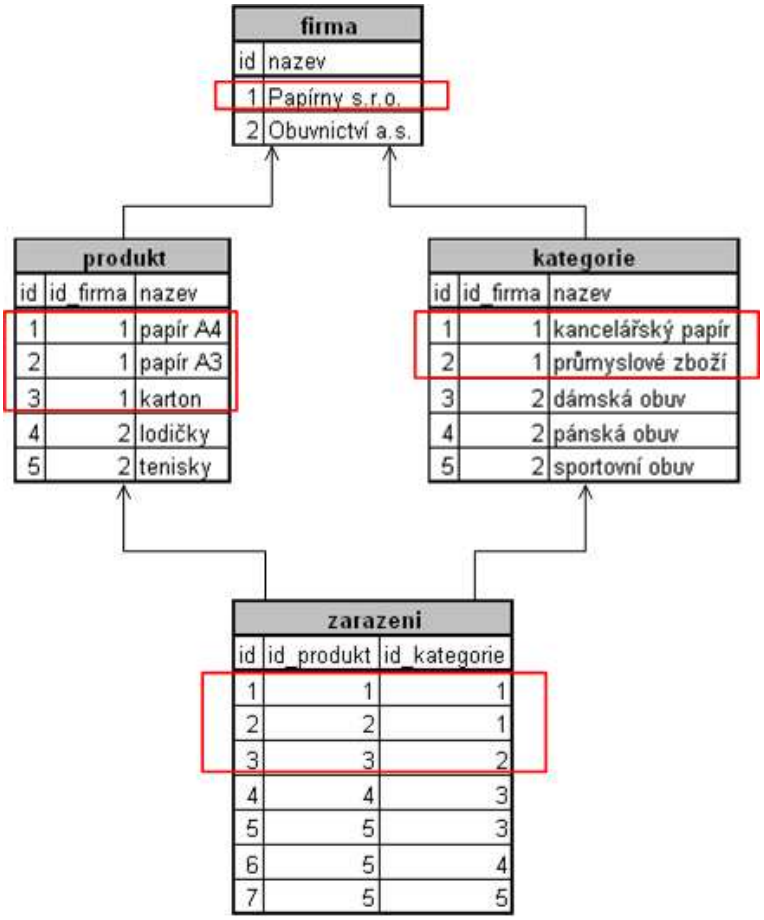
Aktivní vazba na základě množiny dětských kandidátů určuje množinu rodičovských kandidátů.

Běžná vazba na základě množiny rodičovských kandidátů určuje množinu dětských kandidátů.

Rekurzivní vazba je vazba tabulky na sebe samu. Typickým příkladem je například tabulka zaměstnanců s vazbou *nadřízený*.

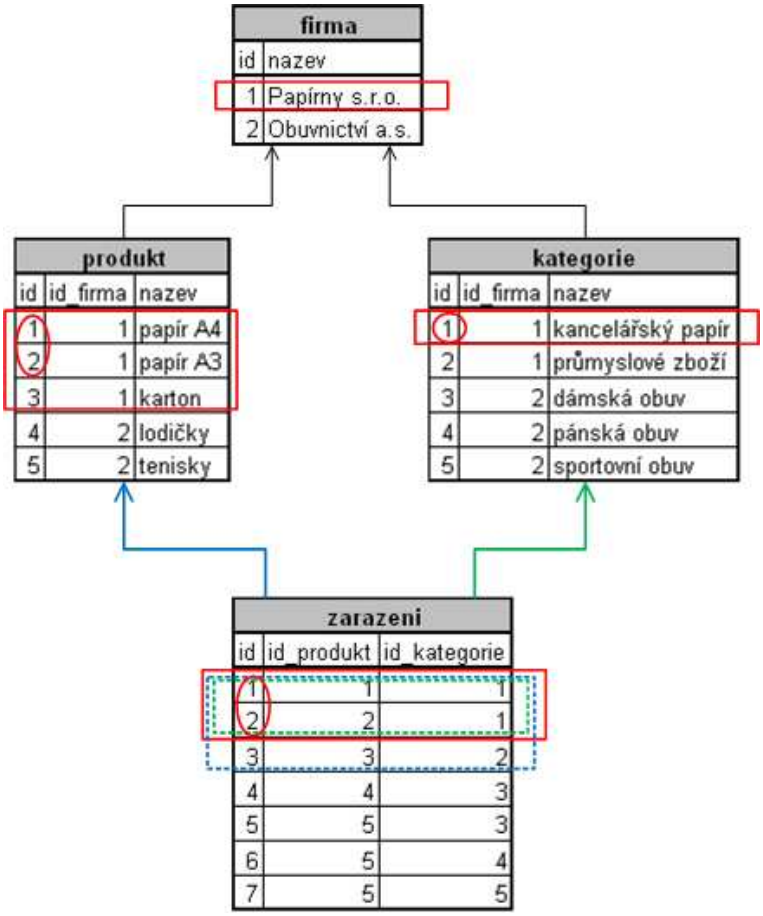
2.3.1 Ilustrace

Pro názornost si nyní uveďme ilustraci k problematice operačních kandidátů, operačních záznamů a odstranitelných operačních záznamů. Na obrázku vidíme příklad pro fiktivní evidenci firem. Hlavní tabulkou je *firma*, operační oblast tvoří všechny 4 tabulky. Chceme provést klonování nebo smazání firmy „Papírny s.r.o.“. Množina kandidátů na operační záznamy, množina operačních záznamů i množina odstranitelných operačních záznamů (pro operaci mazání) jsou shodné. Na obrázku jsou vyznačeny červenými obdélníky.



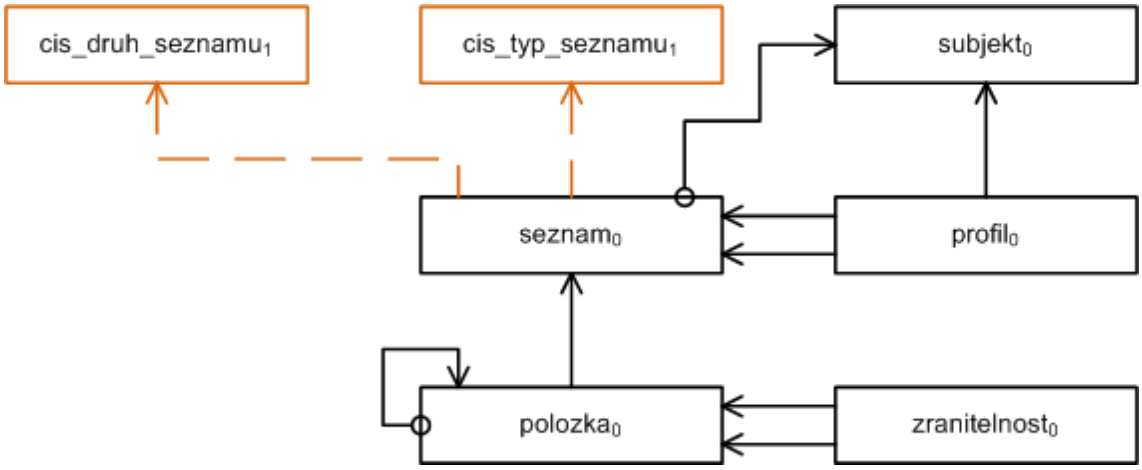
Obrázek 2.4

Na dalším obrázku je znázorněna stejná situace, navíc rozšířená o podmínku, že chceme do operace zahrnout pouze kategorii „kancelářský papír“. Všimněme si, že množina kandidátů a operačních záznamů v tabulce *zarazeni* již není stejná (množiny kandidátů jednotlivých vazeb jsou znázorněny přerušovaným obdélníkem). Rovněž množina odstranitelných operačních záznamů (v tomto obrázku vyznačena elipsou) tabulky *produkt* se liší od množiny jejích operačních záznamů – nemůžeme odstranit produkt „karton“, protože je k němu evidováno zařazení, které nebude odstraněno. Důsledkem toho je, že nemůžeme odstranit ani záznam o firmě – je k ní evidován produkt „karton“, který nebude odstraněn.



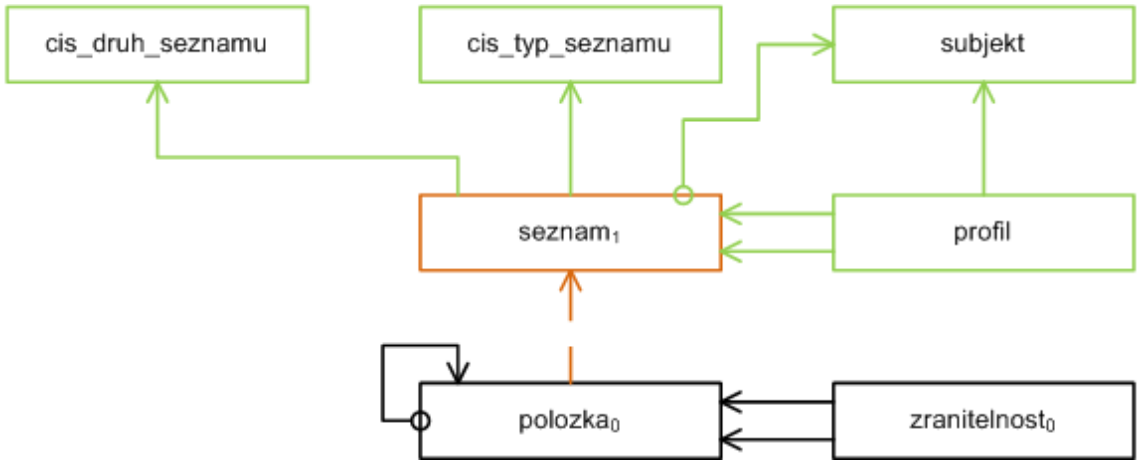
Obrázek 2.5

Dále si názorně ukážeme jednotlivé oblasti, tj. detekovanou oblast, plně detekovanou oblast a operační oblast. Na obrázku vidíme výchozí stav pro operaci nad databází Riskan. Jako hlavní tabulku jsme zvolili *subjekt*. Oranžové tabulky a vazby jsou částečně detekované. Čárkovaně jsou zobrazeny aktivní vazby. Podoblast je znázorněna indexem za názvem tabulky. Všimněme si, že vazby na oba číselníky jsou aktivní. Je tomu tak proto, že operace se k těmto tabulkám (samozřejmě pokud je zařadíme do operační oblasti) nemůže dostat jinak, než od jejich potomků, tedy pouze aktivní vazbou (potomek zde určuje, co bude zpracováno v rodiči). Pokud uživatel změní vazbu na běžnou, znemožní tím zařazení příslušné tabulky do operační oblasti. Podrobnostem o úpravách operační oblasti se věnuje další kapitola (2.4).



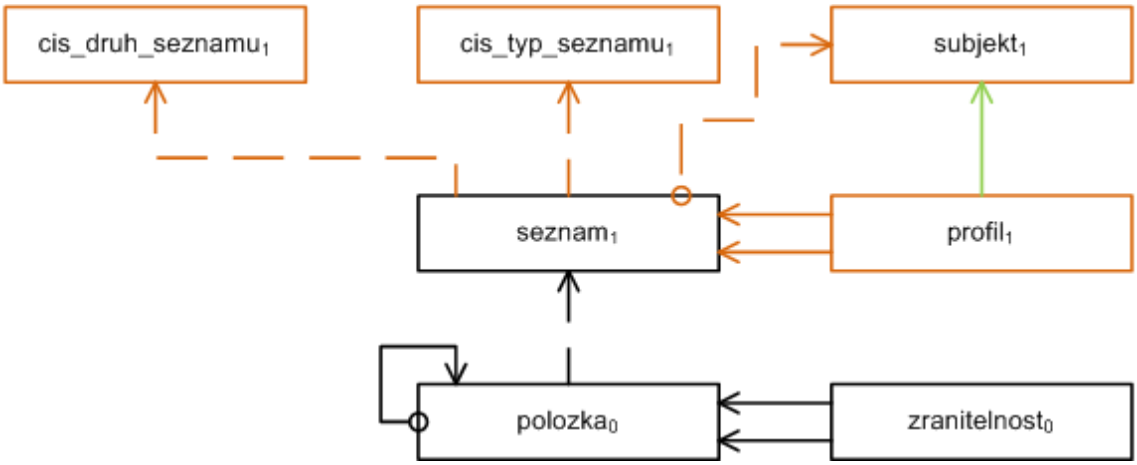
Obrázek 2.6

Následující obrázek zobrazuje výchozí stav nad databází Riskan, pokud jako hlavní tabulku zvolíme tabulku *polozka*. Zelenou barvou jsou zobrazeny tabulky a vazby, o nichž se v tuto chvíli „neví“.



Obrázek 2.7

Na dalším obrázku vidíme stav po rozšíření předchozí situace o tabulku *seznam*.



Obrázek 2.8

2.4 Úpravy operační oblasti

Úpravy operační oblasti spočívají ve vkládání tabulek a vazeb do operační oblasti, případně ve vyjímání tabulek a vazeb z operační oblasti. Vložit do operační oblasti lze pouze plně detekovanou tabulku či vazbu.

2.4.1 Vyjmutí z operační oblasti

Rozlišujeme dva druhy vyjmutí z operační oblasti – implicitní vyjmutí a explicitní vyjmutí.

Explicitním vyjmutím tabulky *A* rozumíme přímé vyjmutí této tabulky uživatelem.

Implicitním vyjmutím tabulky *A* pak rozumíme nepřímé vyjmutí této tabulky, které je důsledkem explicitního vyjmutí jiné tabulky či vazby nebo důsledkem změny typu některé vazby (z běžné na aktivní a naopak).

Obdobně explicitním vyjmutím vazby *X* rozumíme přímé vyjmutí této vazby uživatelem, implicitním vyjmutím pak nepřímé vyjmutí vazby *X* způsobené vyjmutím jiné tabulky či vazby z operace.

Implicitní vyjímání se řídí určitými pravidly. Aby mohla být tabulka *A* součástí operační oblasti, musí platit že

1. není vyjmuta žádná povinná (tj. v cizím klíči neumožňující hodnotu NULL) běžná ani aktivní vazba k rodiči ze stejné podoblasti
2. není-li *A* hlavní tabulkou, musí existovat alespoň jedna nevyjmutá (tj. operační) vazba k rodiči ze stejné podoblasti nebo alespoň jedna aktivní operační vazba od aktivního potomka z libovolné podoblasti

Při nesplnění libovolného z těchto dvou pravidel je tabulka implicitně vyjmuta z operační oblasti.

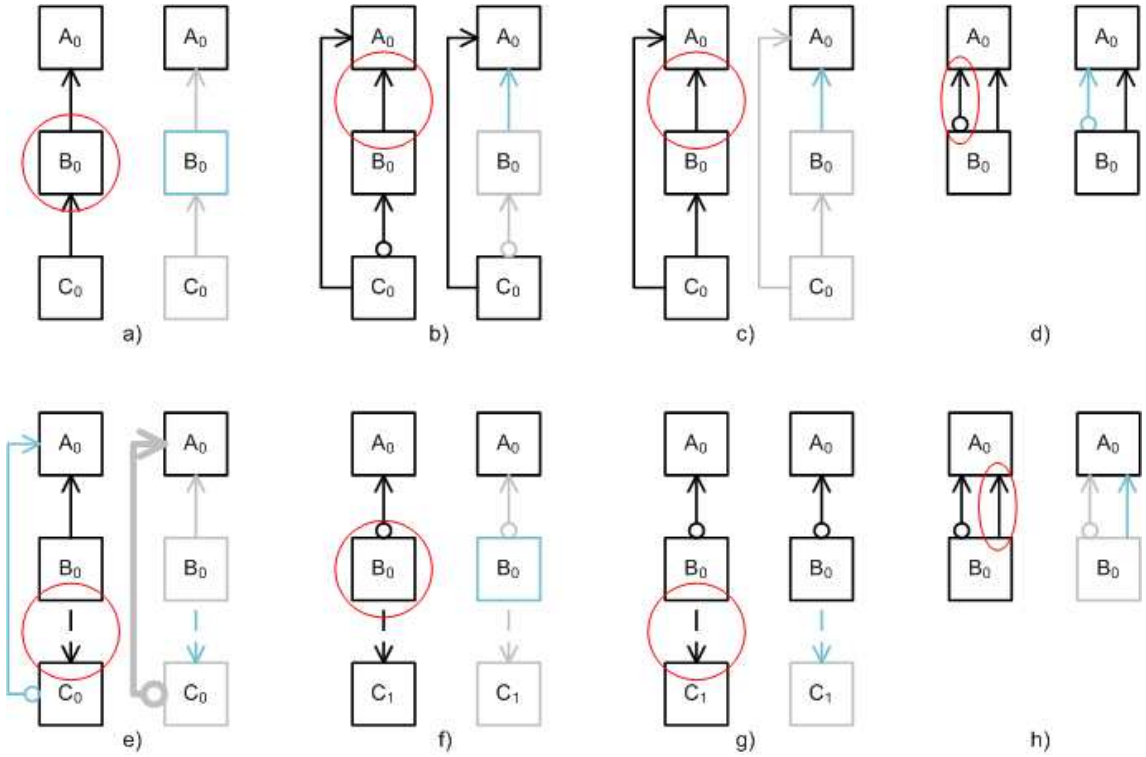
Dále platí, že vyjmutí tabulky *A* způsobí implicitní vyjmutí všech vazeb, jichž je *A* členem, - vazba mezi operační a neoperační tabulkou nemůže být operací korektně zpracována. Dá se tedy říci, že vazba může být součástí operační oblasti, pokud na obou jejich stranách jsou operační tabulky. V opačném případě bude implicitně vyjmuta.

2.4.2 Vložení do operační oblasti

Při vložení tabulky *A* do operační oblasti je kontrolováno, zda je možné pro některou z jejich vazeb zrušit implicitní vyjmutí. Pokud ano, je implicitní vyjmutí zrušeno. Obdobně při vložení vazby *X* do operační oblasti je kontrolováno, zda lze pro některou z vázaných tabulek zrušit implicitní vyjmutí, i zde samozřejmě platí, že pokud ano, tak je zrušeno.

2.4.3 Ilustrace

Pro ilustraci si uveďme několik jednoduchých příkladů. Levá část každého z nich ukazuje stav před vyjmutím tabulky či vazby, pravá pak stav po něm. Hlavní tabulkou je vždy tabulka *A*, explicitně vyjímaná tabulka či vazba je zakroužkovaná. Modrá barva značí explicitní vyjmutí, šedá implicitní vyjmutí. Pokud je jako důsledek provedené úpravy nějaká tabulka či vazba vyjmuta explicitně i implicitně, je zobrazena šedě a tlustší čarou. Pro připomenutí ještě uvádím, že šipka směřuje k tabulce primárního klíče, kolečko označuje cizí klíč umožňující NULL a index u názvu tabulky značí podoblast, do které tabulka patří. Čárkovanou čarou jsou označeny aktivní vazby.



Obrázek 2.9

2.4.4 Praktické využití

V praxi využijeme vyjímání z operační oblasti tehdy, kdy výchozí operační oblast zahrnuje pro nás zbytečně mnoho tabulek. Například chci vytvořit klon firmy a jejich výrobků – v tom případě je pro mne ale zbytečné, abych klonoval i kategorie a zařazení výrobků do kategorií. Proto vyjmu tabulku kategorií z operační oblasti, tabulka zařazení bude vyjmuta (dle výše uvedených pravidel) implicitně.

Příkladem využití vkládání do operační oblasti může být situace, kdy chceme naklonovat některé záznamy z tabulky *produkt* (tu zvolíme jako hlavní tabulku) s tím, že ke každému produktu chceme vytvořit i klon firmy, která jej vyrábí. Rozšíříme plně detekovanou oblast o tabulku *firma*, ta bude automaticky ihned zařazena do operační oblasti (protože vyhovuje podmínkám pro zařazení do operační oblasti). Vazba mezi tabulkami *produkt* a *firma* samozřejmě bude aktivní – produkty určují, které firmy budu klonovat, ne naopak.

2.5 Možnosti využití aktivních vazeb

Aktivní vazby jsou, kromě již uvedeného využití pro rozšíření operační oblasti, využitelné také v rámci stejné podoblasti. Dobrým příkladem může být právě databáze Riskan, konkrétně tabulka *profil*.

Jak je uvedeno v úvodním popisu databáze Riskan, tabulka profilů obsahuje profily patřící konkrétnímu subjektu. Profil využívá dva seznamy – seznam aktiv a seznam hrozeb. Druh těchto seznamů je „použitý seznam“, který nemá vazbu na subjekt – cizí klíč *id_subjekt* má hodnotu NULL.

Uživatel chce provést operaci – předpokládejme klonování – vybraného subjektu. Vybere tedy tabulku *subjekt* jako hlavní tabulku, nastaví omezující podmínku na konkrétní subjekt a spustí operaci. Výsledkem této konfigurace mimo jiné vždy bude 0 operačních záznamů v tabulce *profil*. Proč tomu tak je? Profil je vázán dvěma povinnými vazbami na tabulku subjektů. Jenže ani jedna z těchto vazeb negeneruje žádného operačního kandidáta tabulky *profil*, protože žádný ze seznamů použitých v profilu není vázán na subjekt.

Zde se dostávají ke slovu aktivní vazby. Změníme-li typ obou vazeb mezi tabulkami *subjekt* a *profil* z běžných na aktivní, v podstatě tím řekneme, že chceme (mimo jiné) vytvořit klon všech profilů subjektu s tím, že ke každému klonovanému profilu chceme vytvořit klon jeho seznamu aktiv a klon jeho seznamu hrozeb.

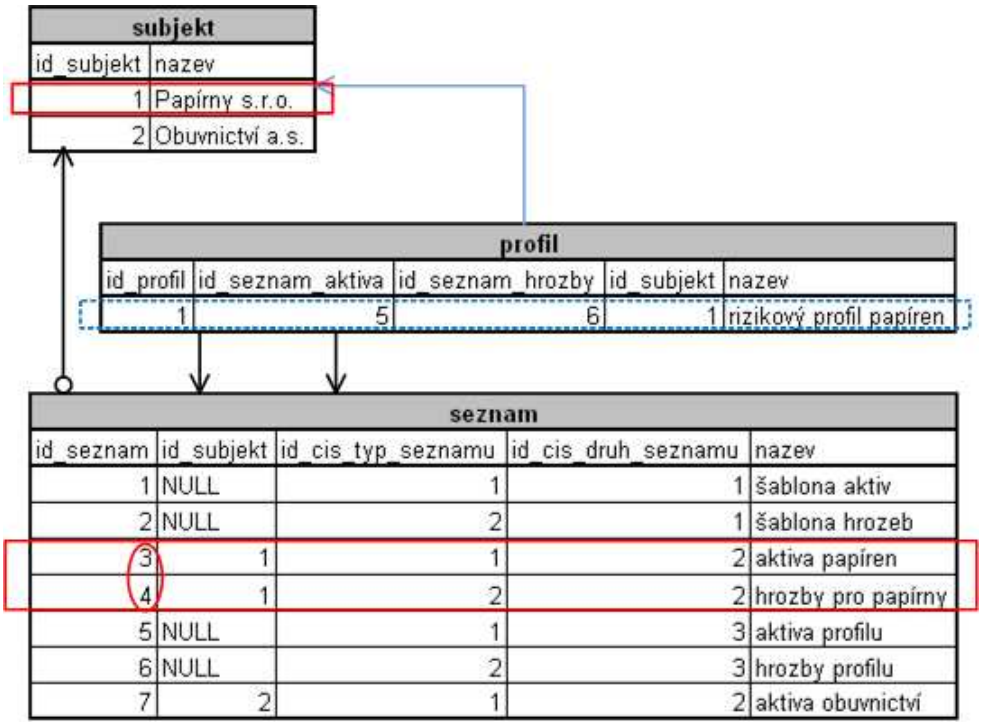
V obou případech pak vazba mezi tabulkami *subjekt* a *seznam* bude klonovat pouze seznamy typu „vlastní seznam“, které se vážou na subjekt.

Obecně nejspíše nelze stanovit závazná pravidla pro použití aktivních vazeb v rámci stejné podoblasti. Záležet bude vždy na konkrétním způsobu a významu uložených dat. Uvedený příklad se však může stát vodítkem.

2.5.1 Ilustrace

Opět pro snadnější pochopení přikládám ilustraci uvedeného příkladu nad databází Riskan. Na obrázku nejsou pro přehlednost zobrazeny číselníky, tabulka *polozka* ani tabulka *zranitelnost*. Operační záznamy jsou ohraničeny plným obdélníkem, odstranitelné operační záznamy elipsou.

První obrázek zobrazuje situaci při běžných vazbách.



subjekt

id_subjekt	nazev
1	Papírny s.r.o.
2	Obuvnictví a.s.

profil

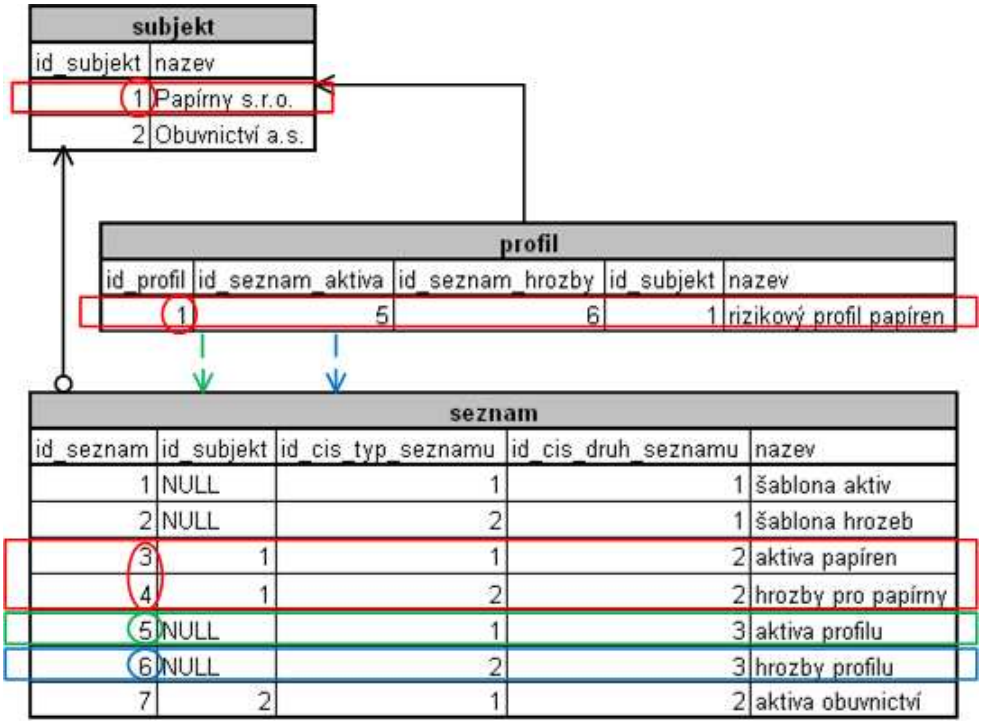
id_profil	id_seznam_aktiva	id_seznam_hrozby	id_subjekt	nazev
1	5	6	1	rizikový profil papíren

seznam

id_seznam	id_subjekt	id_cis_typ_seznamu	id_cis_druh_seznamu	nazev
1	NULL	1	1	šablona aktiv
2	NULL	2	1	šablona hrozeb
3	1	1	2	aktiva papíren
4	1	2	2	hrozby pro papírny
5	NULL	1	3	aktiva profilu
6	NULL	2	3	hrozby profilu
7	2	1	2	aktiva obuvnictví

Obrázek 2.10

Druhý obrázek zobrazuje situaci při použití aktivních vazeb mezi tabulkami *profil* a *seznam*.



subjekt

id_subjekt	nazev
1	Papírny s.r.o.
2	Obuvnictví a.s.

profil

id_profil	id_seznam_aktiva	id_seznam_hrozby	id_subjekt	nazev
1	5	6	1	rizikový profil papíren

seznam

id_seznam	id_subjekt	id_cis_typ_seznamu	id_cis_druh_seznamu	nazev
1	NULL	1	1	šablona aktiv
2	NULL	2	1	šablona hrozeb
3	1	1	2	aktiva papíren
4	1	2	2	hrozby pro papírny
5	NULL	1	3	aktiva profilu
6	NULL	2	3	hrozby profilu
7	2	1	2	aktiva obuvnictví

Obrázek 2.11

2.6 Rekurzivní tabulky

Rekurzivním tabulkám (někdy také označovaným jako hierarchické tabulky) je věnována zvláštní pozornost. Kromě běžného zpracování (kdy vazby k rodičům definují každá svou množinu operačních kandidátů a z nichž jsou určeny operační záznamy), umožňuje nástroj také tzv. pokročilé zpracování.

Pokročilé zpracování upravuje způsob detekce operačních kandidátů a způsob aplikování omezující podmínky.

Nástroj zavádí dvě volby pokročilého zpracování

3. zachování 1. úrovně
4. aplikace omezující podmínky jen na 1. úroveň

2.6.1 Zachování 1. úrovně

Volba zaručuje, že operačními záznamy se nestanou žádné záznamy, které by v rámci operačních záznamů tvořily 1. úroveň hierarchie, přičemž v rámci operačních kandidátů první úrovní nejsou. Jinými slovy – pokud se operační kandidát 1. úrovně nestane operačním záznamem, nestanou se jimi žádní jemu podřízení kandidáti.

Ostatní úrovně speciální zpracování nepotřebují – pokud není jim nadřazený záznam nalezen v operačních záznamech, nestanou se ani ony operačními záznamy.

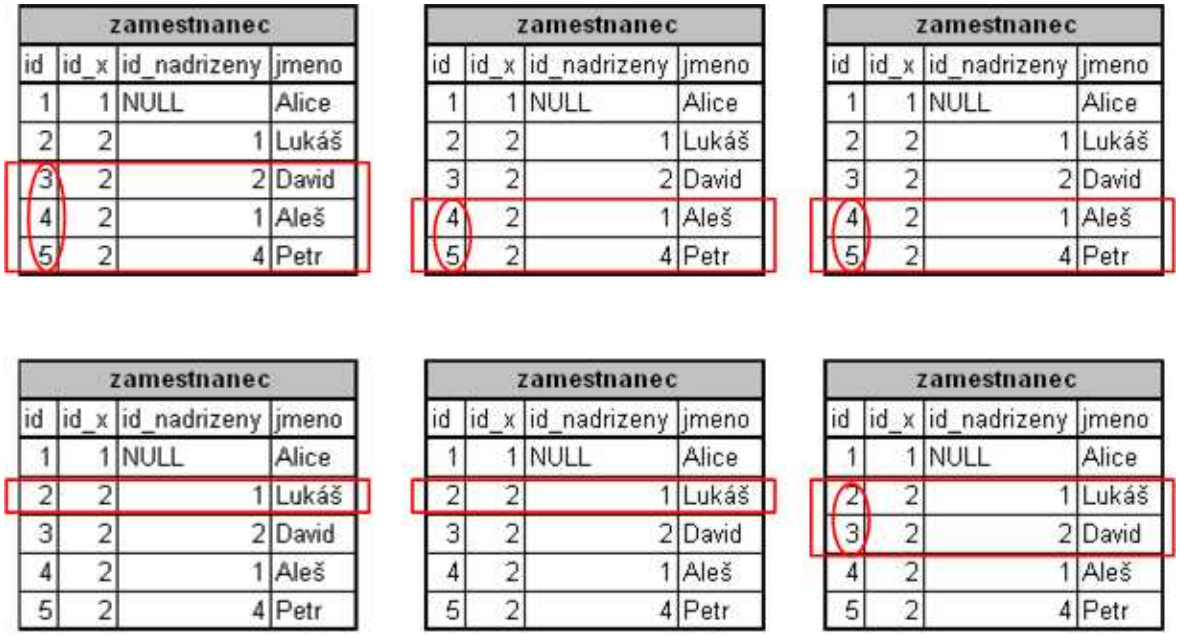
2.6.2 Aplikace omezující podmínky jen na 1. úroveň

Pro použití této možnosti je nutné použít volbu zachování 1. úrovně.

Volba zúží aplikaci omezující podmínky pouze na operační kandidáty 1. úrovně.

2.6.3 Ilustrace

Na obrázku vidíme tabulku zaměstnanců. Vlevo je situace bez zapnuté volby pro zachování 1. úrovně. Uprostřed je volba zapnutá, vpravo je pak navíc zvolena aplikace podmínky jen na 1. úroveň. Omezující podmínka je v prvním řádku `jmeno <> 'Lukáš'`, v druhém řádku `jmeno = 'Lukáš'` Cizí klíč *id_x* představuje vazbu do nějaké rodičovské operační tabulky, předpokládejme, že kandidáty této vazby jsou záznamy s hodnotou 2, tedy všechny vyjma záznamu o Alici. Ta je nadřízená všech ostatních zaměstnanců – je v tabulce zaměstnanců záznamem 1. úrovně. V rámci operačních kandidátů jsou však na 1. úrovni záznamy o Lukášovi a Alešovi. Nás zajímá právě první úroveň operačních kandidátů.



Obrázek 2.12

3 Operační algoritmy

V této části budou popsány oba operační algoritmy, tj. algoritmus pro klonování data a algoritmus pro mazání dat. K popisu jsou použity vývojové diagramy a příslušný textový popis. Není popisována detailní implementace algoritmu, ale jeho funkce. V diagramech a některých textových popisech se vyskytuje pseudokód podobný jazyku C#.

3.1 Obecná charakteristika algoritmů

Oba algoritmy pracují nad operační oblastí. Zpracování začíná u hlavní tabulky a pokračuje rekurzivně dále.

Algoritmy nezjišťují před spuštěním konkrétní pořadí zpracování tabulek a vazeb v operační oblasti.

Zjednodušeně řečeno algoritmy před vlastním provedením operace nad konkrétní tabulkou detekují kandidáty na operační záznamy. Po ukončení detekce kandidátů jsou z nich odstraněny kandidáti, kteří nemohou být operačními záznamy. Zůstane tak množina operačních záznamů, nad nimiž konečně dojde k samotnému provedení operace.

3.1.1 Úložiště operačních kandidátů a operačních záznamů

Záznamy, které jsou operačními kandidáty, musí být vhodným způsobem označeny. U operace klonování navíc musí být u operačních záznamů evidována nová hodnota primárního klíče a nové hodnoty operačních cizích klíčů.

Otázka zní, co je oním vhodným způsobem. Zvažoval jsem dvě možnosti. První spočívala v rozšíření operačních tabulek o pomocné sloupce. Druhá spočívala ve využití dočasných tabulek. K této variantě jsem se nakonec přiklonil a to z těchto důvodů

1. není zasahováno do struktury operačních tabulek, což je (mimo jiné) obrovská výhoda v případě selhání operace (např. z důvodu ztráty připojení k databázi) – v tabulkách nezůstávají pomocné sloupce
2. dočasné tabulky jsou odstraněny při ukončení spojení, které je vytvořilo, což je opět výhoda v případě havárie operace

Konkrétní obsah dočasných tabulek je pro klonování a mazání rozdílný a budu se mu věnovat při popisu jednotlivých algoritmů.

Odstraňování operačních kandidátů

Operační kandidáti, u kterých je zjištěno, že se nemohou stát operačními záznamy, jsou v některých místech algoritmu z dočasné tabulky odstraňovány.

3.2 Omezení kladená na operační oblast

Na operační oblast jsou kladená určitá omezení, která jsou dána vyvinutými algoritmy.

3.2.1 Primární klíče

Každá tabulka v detekované oblasti musí mít definován primární klíč. V opačném případě je zobrazeno chybové hlášení.

Primární klíče na více sloupcích jsou podporovány.

3.2.2 Cizí klíče

Nejsou podporovány cizí klíče obsahující identity (automatické číslování, též známo jako autoincrement) sloupce.

Není také možné používat nástroj tam, kde jsou definovány nad jedním sloupcem dva a více cizích klíčů.

Cizí klíče na více sloupcích jsou podporovány.

3.2.3 Vazby

Nástroj podporuje pouze vazby mezi primárním a cizím klíčem. Některé databázové systémy (MS SQL mezi ně patří) podporují také vazby mezi unikátním constraint (omezením) a cizím klíčem, nástroj je ovšem ignoruje.

3.2.4 Cyklické vazby

Vyjma rekurzivních vazeb nejsou podporovány cyklické vazby, tj. například situace, kdy tabulka *A* je rodičem tabulky *B* která je rodičem tabulky *A.* Teoreticky podpora možná je, ovšem prakticky při hledání způsobu, jak podpory docílit, jsem došel k závěru, že poměr vynaložené úsilí / užitná hodnota by byl velice nevýhodný. Jinými slovy – implementovat podporu cyklických vazeb by bylo neúměrně náročné vzhledem k praktickému využití této funkcionality. To je totiž dle mého názoru mizivé; pátrání po praktickém využití nepřineslo žádný výsledek.

Kromě případné definice cyklické vazby přímo v databázi je možné vytvořit cyklickou vazbu také nevhodnou kombinací aktivních a běžných vazeb. Ani tato situace není podporována.

Pokud je detekován cyklus definovaný přímo v databázi, libovolná kombinace aktivních i běžných vazeb patřících do tohoto cyklu bude stále považována za cyklus.

3.3 Pomocné metody

Oba algoritmy využívají pomocné metody pro detekci neoperačních záznamů a pro pokročilé zpracování rekurzivních tabulek. Klonování pak navíc využívá metodu pro nastavení klonovaných hodnot primárních klíčů.

Poznámka: v reálné implementaci se metody pro jednotlivé operace lehce odlišují a každá operace má svou vlastní implementaci těchto metod. Zde si je však dovolím sjednotit.

3.3.1 Detekce neoperačních záznamů

Metoda *Inoperable* detekuje operační kandidáty, kteří se nemohou stát operačními záznamy, tj. nejsou operačními kandidáty všech běžných vazeb k rodičům a

- všech rekurzivních vazeb při nastavení parametru *includeRecursiveFKs* na true
- všech aktivních vazeb k rodičům při nastavení parametru *includeActiveFKs* na true

Pokud je některá z těchto vazeb nepovinná, považuje se příslušný operační kandidát za operačního kandidáta vazby tehdy, když má související záznam tabulky v příslušném cizím klíči hodnotu NULL.

Detekovaným neoperačním záznamům je nastaven příznak *_deletable* na hodnotu 1. V případě volání metody s parametrem *delete* nastaveným na true dojde vzápětí k odstranění těchto záznamů z tabulky kandidátů.

3.3.2 Pokročilé zpracování rekurzivních tabulek

O pokročilé zpracování rekurzivních tabulek se stará metoda *AdvancedRecursive*. Pokud je volána operací klonování nad tabulkou v operačním stavu 2, jsou nejprve naplněny hodnoty primárních klíčů.

Pokud je nastaven parametr metody *ApplyWhere* na true, je aplikována omezující podmínka a to v závislosti na příslušném nastavení operační tabulky buď na všechny záznamy, nebo pouze na první úroveň. Kandidátům nevyhovujícím podmínce je nastaven příznak *_delete* na 1.

Následuje propagace hodnoty *_delete* == 1 příslušných záznamů k potomkům těchto záznamů, čímž je zajištěno zachování úrovní. Poznamenejme, že hodnotu *_delete == 1* mohly mít někteří operační kandidáti nastavenu ještě před voláním metody *AdvancedRecursive* (proto je propagace prováděna vždy a ne jen při aplikaci omezující podmínky).

Posledním krokem je odstranění kandidátů s *_delete == 1*.

3.3.3 Klonované hodnoty primárního klíče

Operačním záznamům je nutné nastavit nové hodnoty primárního klíče. O to se stará metoda *ClonePKValues*.

Pokud primární klíč obsahuje identity sloupec, je tomuto sloupci v tabulce kandidátů nastavena hodnota vypočítaná na základě poslední identity hodnoty (MS SQL funkce ident_current('[název tabulky]'), k níž je připočítáno pořadové číslo záznamu v tabulce kandidátů vynásobeno velikostí přírůstku identity sloupce.

V případě, že primární klíč neobsahuje identity sloupec, ale obsahuje sloupec typu *uniqueidentifier* (GUID – Global Unique Identifier), je do tohoto sloupce vygenerováno GUID pomocí MS SQL funkce newid().

Pokud primární klíč neobsahuje ani identity ani uniqueidentifier sloupec, je nutné, aby funkci pro stanovení nové hodnoty primárního klíče zadal uživatel.

3.4 Algoritmus klonování dat

Přejděme nyní k samotnému algoritmu klonování dat.

Klonování je realizováno metodou

```
bool? Clone(table, opRship)
```

Parametr *table* je aktuálně zpracovávaná operační tabulka, parametr *opRship* je vazba, přes která zpracování spustila. Při prvním volání je vazba NULL, tabulka je nastavena na hlavní tabulku.

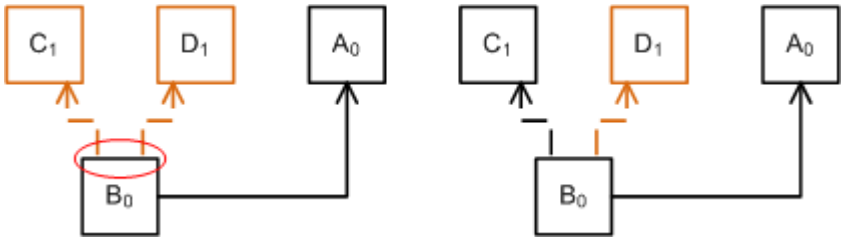
3.4.1 Unikátní indexy

Unikátní indexy představují potenciální problém pro klonování záznamů. Existují dvě problematické situace.

První nastane, když unikátní index nezahrnuje cizí ani primární klíč. Klonem záznamu pak bude porušena unikátnost hodnot v indexu. Řešením této situace je uživatelská modifikace *Insert Value*, tedy vkládané hodnoty sloupce.

Druhou situací je, když unikátní index obsahuje pouze neklonované cizí klíče. Hodnota neklonovaného cizího klíče je v klonu záznamu stejná, i zde tedy dojde k porušení unikátnosti hodnot v indexu. Uživatelská modifikace *Insert Value* zde nepřichází v úvahu – došlo by k porušení referenční integrity. Jediným řešením – nepovažujeme-li vyjmutí tabulky z operační oblasti za řešení – je vložení alespoň jedné vazby, jejíž cizí klíč je součástí unikátního indexu, do operační oblasti.

Uveďme si ilustrační příklad. Hlavní tabulkou je tabulka *A*, tabulka *B* má unikátní index na vazbách k tabulkám *C* a *D*. Ty Aby byla zajištěna unikátnost indexu v tabulce *B*, musí uživatel vložit do operační oblasti alespoň jednu z tabulek *C* a *D* (situace vpravo).



Obrázek 3.1

3.4.2 Řízení operace

Algoritmus ke každé tabulce eviduje stav. Tabulka v průběhu zpracování přejde postupně 8 různými stavy:

0. nezpracovaná
1. zpracování běžných vazeb k rodičům a aktivních vazeb od potomků
2. zpracování rekurzivních vazeb
3. zpracování aktivních vazeb k rodičům
4. kontrola úplného zpracování aktivních vazeb k rodičům
5. vlastní vykonání operace, aktualizace cizích klíčů pasivních potomků
6. zpracování běžných vazeb k potomkům
7. zpracovaná

Dále je třeba evidovat stav k aktivním vazbám:

0. nezpracovaná
1. zpracovaná
2. částečně zpracovaná (vazbou byli detekováni operační kandidáti, ale cizí klíč vazby dosud nebyl aktualizován klonovanou hodnotou primárního klíče)

3.4.3 Obsah dočasné tabulky

Dočasná tabulka pro operaci klonování obsahuje

- sloupce primárního klíče
- sloupce *clone_[název sloupce primárního klíče]* pro uložení nové hodnoty primárního klíče
- sloupce všech operačních cizích klíčů
- při pokročilém zpracování rekurzivních vazeb sloupec *#[název vazby]* uchovávající příznak, zda je kandidát první úrovní (hodnota 1)
- sloupec *_active* pro uchování příznaku, zda byl kandidát, případně jeho rodič či aktivní potomek, vložen do operace aktivní vazbou (hodnota 1)
- sloupec *_delete* pro uchování příznaku, zda má být operační kandidát odstraněn z tabulky kandidátů, tj. kandidát se nemůže stát operačním záznamem (hodnota 1)
- sloupec *_cloned* pro uchování příznaku, zda byl záznam naklonován (hodnota 1)

3.4.4 Význam proměnných a vlastností v diagramech

V diagramech se vyskytují některé proměnné a vlastnosti tabulek a vazeb. Zde je jejich přehled.

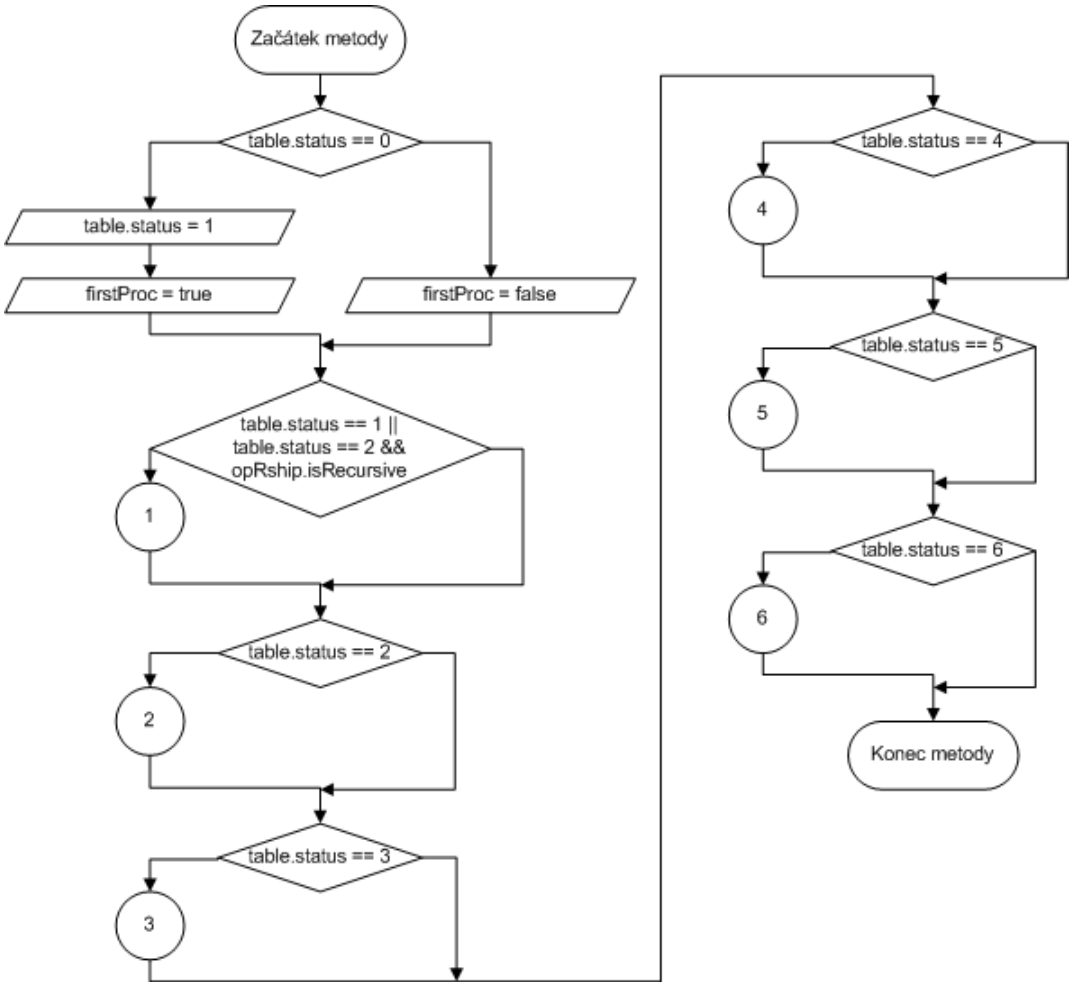
table.status	status tabulky
table.norm	běžné vazby k rodičům
table.normCnt	počet běžných vazeb k rodičům
table.normProc	počet zpracovaných běžných vazeb k rodičům
table.pass	aktivní vazby od potomků
table.passCnt	počet aktivních vazeb od potomků
table.passProc	počet zpracovaných aktivních vazeb od potomků
table.act	aktivní vazby k rodičům
table.actCnt	počet aktivních vazeb k rodičům
table.rec	rekurzivní vazby
table.recCnt	počet rekurzivních vazeb
table.isMain	příznak, zda jde o hlavní tabulku
table.preserve1st	příznak, zda budou zachovány 1. úrovně rekurzivních vazeb
opRship.status	status vazby
opRship.isRecursive	příznak, zda jde o rekurzivní vazbu
opRship.isActive	příznak, zda jde o aktivní vazbu
opRship.activeProc	počet vyvolaných zpracování aktivní vazby
opRship.activeProcCnt	počet teoretického maximálního počtu volání zpracování aktivní vazby
firstProc	příznak, zda jde o první zpracování tabulky (tj. tabulka ještě nebyla zpracována žádnou operační vazbou)

Tabulka 3.1

3.4.5 Popis algoritmu

Poznámka: levá větev rozhodovacího bloku v diagramech značí splnění podmínky, pravá pak nesplnění podmínky.

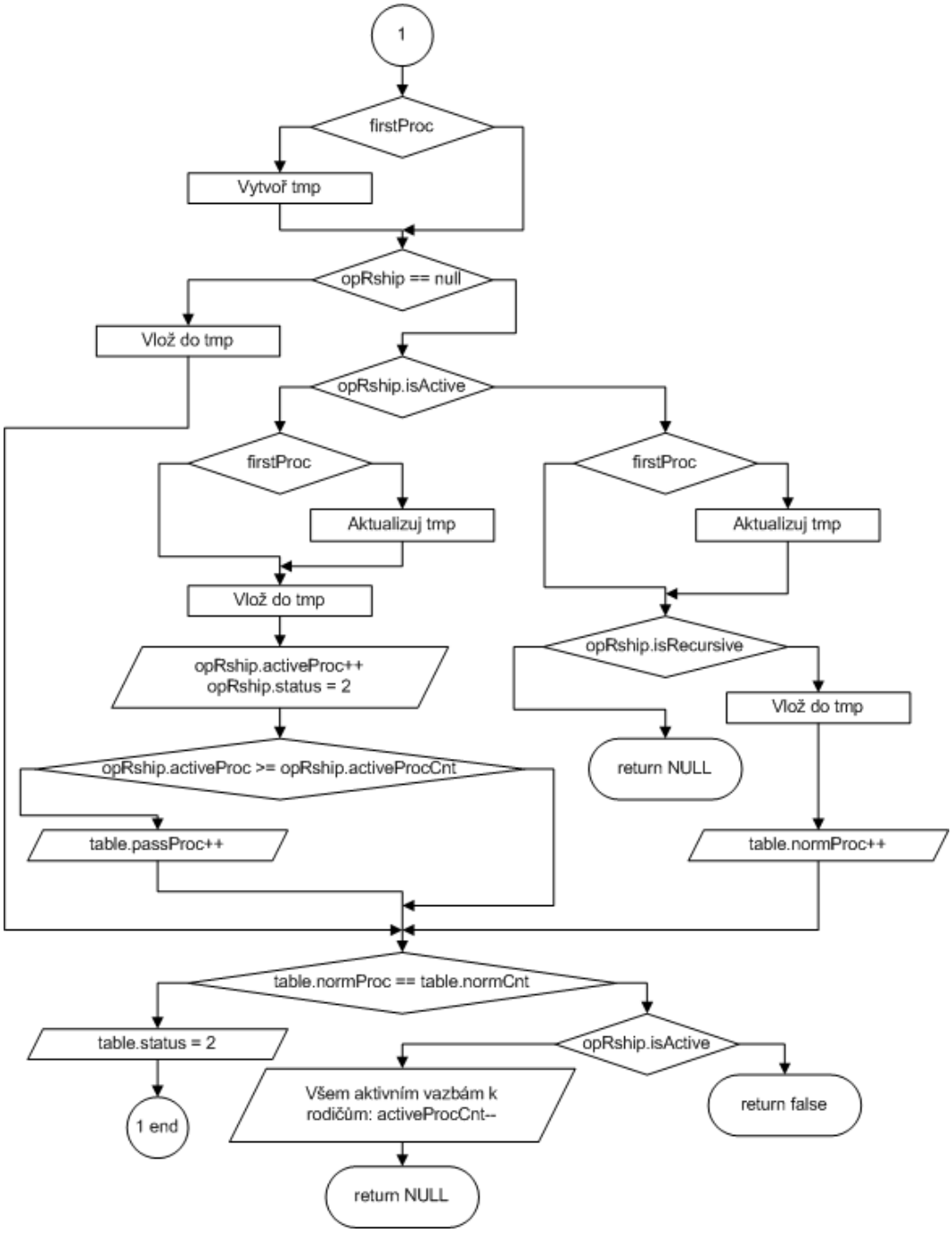
Kostra metody Clone



Obrázek 3.2

Zpracování stavu 1

Pro tabulku ve stavu 1 jsou detekováni operační kandidáti jednotlivých vazeb. Tato část metody *Clone* je rovněž využívána pro zpracování rekurzivních vazeb u tabulek ve stavu 2.



Obrázek 3.3

Jedná-li se o první zpracování tabulky *table*, je vytvořena dočasná tabulka operačních kandidátů. Následuje samotná detekce kandidátů.

Pokud je *opRship* NULL, jde o první zpracování hlavní tabulky a stačí tak provést pouze vložení operačních kandidátů, zde konkrétně hodnot primárních klíčů těch záznamů tabulky *table*, které vyhovují příslušným omezujícím podmínkám.

Pokud je vazba *opRship* aktivní a nejedná se o první zpracování tabulky, již existujícím kandidátům je nastaven příznak *_active* na hodnotu 1. Následuje samotné vložení kandidátů aktivní vazbou, tj. vložení hodnot příslušných primárních klíčů. Příznak *_active* je u těchto kandidátů nastaven na hodnotu 1. Čítač zpracování aktivní vazby je zvýšen o jedno, pokud bylo dosaženo maximálního teoretického počtu zpracování aktivní vazby, je tabulce zvýšen čítač zpracování aktivních vazeb od potomků.

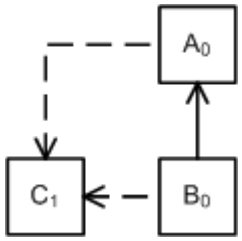
V případě, že *opRship* není aktivní a nejedná se o první zpracování tabulky, již existujícím kandidátům jsou nastaveny hodnoty sloupců příslušného cizího klíče na klonované hodnoty primárních klíčů rodičovské tabulky – tím je zajištěno provázání kopie potomka na kopii rodiče. Další zpracování závisí na tom, je-li vazba rekurzivní. Pokud ano, je metoda ukončena s návratovou hodnotu NULL. Naopak u nerekurzivní vazby jsou vloženi kandidáty této vazby, tj. primární klíče příslušných záznamů tabulky *table*. Sloupce cizích klíčů těchto kandidátů jsou nastaveny na klonované hodnoty primárních klíčů rodičovské tabulky. Následuje aktualizace čítače zpracovaných běžných vazeb od rodičů.

Po ukončení samotné detekce závisí další postup na míře zpracování tabulky. Jsou-li zpracovány všechny běžné operační vazby k rodičům, tabulka přejde do stavu 2. V opačném případě, pokud je *opRship* aktivní vazbou, je všem aktivním vazbám k rodičům snížen teoretický maximální počet jejich zpracování (viz. dále) a metoda je ukončena s návratovou hodnotou NULL. Pokud je *opRship* běžná vazba, je návratová hodnota false. To signalizuje vyšší úrovni rekurze (konkrétně zpracování běžných vazeb k potomkům, které je prováděno ve stavu 5) že tabulka *table* nebyla tímto voláním úplně zpracována a bude nutné se k ní ještě vrátit.

Podrobněji k aktivním vazbám

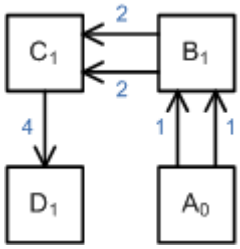
Zastavme se na chvíli u teoretického maximálního počtu zpracování aktivních vazeb (počet označen jako teoretický, protože za běhu algoritmu může dojít k jeho snížení, jak vidíme výše).

Každé zpracování tabulky aktivní vazbou způsobí – pokud jsou již zpracovány všechny běžné vazby od rodičů – okamžité zpracování jejích aktivních vazeb k rodičům, stanovení a klonování dosud neklonovaných (příznak *_cloned* = 0 v tabulce kandidátů) operačních záznamů. Tento postup je nutný, protože nelze čekat se zpracováním až do doby, než budou zpracovány všechny aktivní vazby od potomků – mohlo by dojít k uvíznutí algoritmu. Například pro situaci na obrázku by algoritmus uvízl hned při zpracování hlavní tabulky *A*, která před zpracováním vazby *A*-*B* musí počkat na úplné zpracování vazby *A*-*C*, tedy na klon touto vazbou detekovaných operačních záznamů v tabulce *C*. Ten by však nemohl být proveden před zpracováním vazby *B*-*C*, která závisí na vazbě *A*-*B*. Máme uvíznutí… Při okamžitém klonu aktivní vazbou *A*-*C* detekovaných operačních záznamů k ničemu takovému nedojde.



Obrázek 3.4

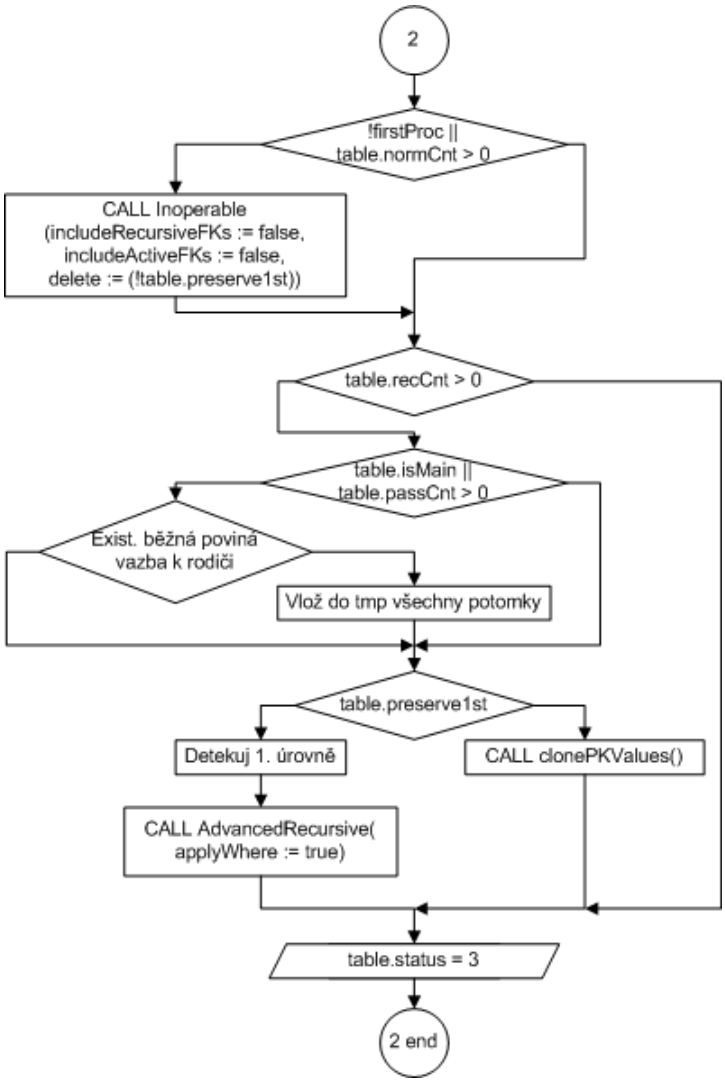
Z hlediska množiny operačních záznamů takový postup nevadí, protože množiny jednotlivých aktivních vazeb od potomků se nemohou vzájemně ovlivňovat (v předchozím případě konkrétně množina vazby *A*-*C* nemůže nijak omezit množinu vazby *B*-*C*). Takové zpracování však způsobí, že aktivní vazba nemusí být zpracována pouze jednou, ale může být zpracována až *n*-krát, kde *n* je součet maximálního počtu zpracování aktivních vazeb všech potomků, od nichž vedou do tabulky aktivní vazby. Viz. obrázek – čísla u vazeb jsou *n*. Nemůžeme tak jednoduše prohlásit aktivní vazbu za zpracovanou hned při jejím prvním zpracování.



Obrázek 3.5

Zpracování stavu 2

Tabulka ve stavu 2 zpracovává své rekurzivní vazby.



Obrázek 3.6

Není-li aktuální zpracování tabulky *table* jejím prvním zpracováním a má-li tabulka běžné operační vazby k rodičům, mohou se mezi operačními kandidáty vyskytovat takoví, kteří se nemohou stát operačními záznamy. Proto zavoláme detekci těchto kandidátů s tím, že pokud není nastaveno pokročilé zpracování rekurzivní tabulky, můžeme tyto kandidáty rovnou odstranit. U pokročilého zpracování je budeme ještě potřebovat. Tato detekce probíhá nezávisle na tom, zda tabulka má či nemá rekurzivní vazby.

V dalším kroku jsou vkládáni mezi kandidáty potomci již detekovaných kandidátů. Podmínkou ovšem je, že tabulka *table* je hlavní tabulkou nebo k ní existuje aktivní vazba od potomka (v tom případě vkládáme pouze potomky kandidátů s příznakem *_active* = 1); u ostatních tabulek je předpokládáno, že všichni operační kandidáti byli vloženi běžnými operačními vazbami. Dále nesmí existovat žádná povinná běžná operační vazba – pokud existuje, nemá smysl pokoušet se o detekci dalších kandidátů. I kdyby byli nějací nalezeni, nemohou se stát operačními záznamy, protože nejsou operačními kandidáty některé z povinných vazeb.

Pokračujeme v závislosti na nastavení pokročilého zpracování. Běžné zpracování vede k zavolání pomocné metody plnící klonované hodnoty primárních klíčů. U pokročilého zpracování jsou nejprve detekovány první úrovně kandidátů a následně je volána pomocná metoda pokročilého zpracování rekurzivních tabulek.

Nakonec přejde tabulka do stavu 3.

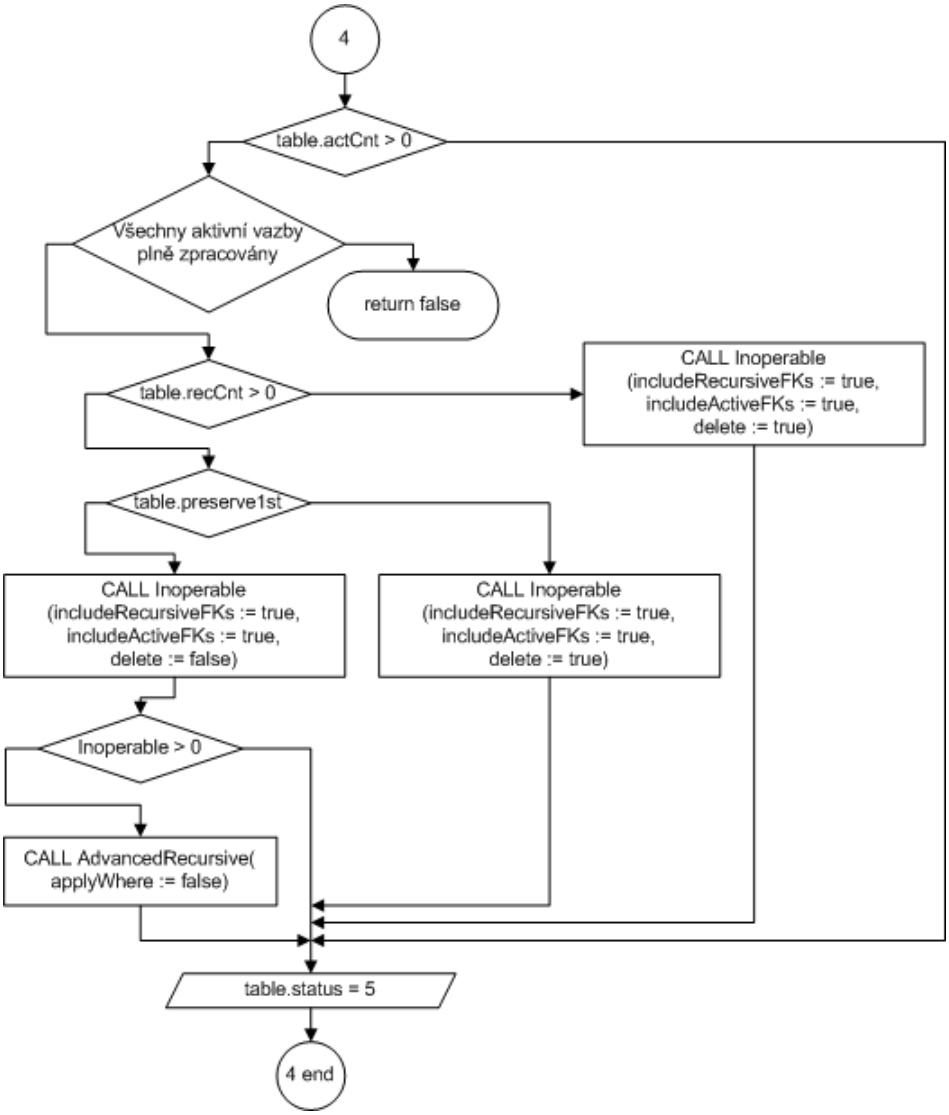
Zpracování stavu 3

Ve stavu 3 je vyvoláno zpracování aktivních vazeb k rodičům. Nakonec přejde tabulka do stavu 4.

Obrázek 3.7

Zpracování stavu 4

Ve stavu 4 je kontrolováno, zda byly všechny aktivní vazby tabulky *table* k rodičům úplně zpracovány, tj. zda již aktualizovali v tabulce kandidátů hodnotu příslušného cizího klíče.



Obrázek 3.8

Pokud se tak nestalo, je metoda *Clone* ukončena s návratovou hodnotou false.

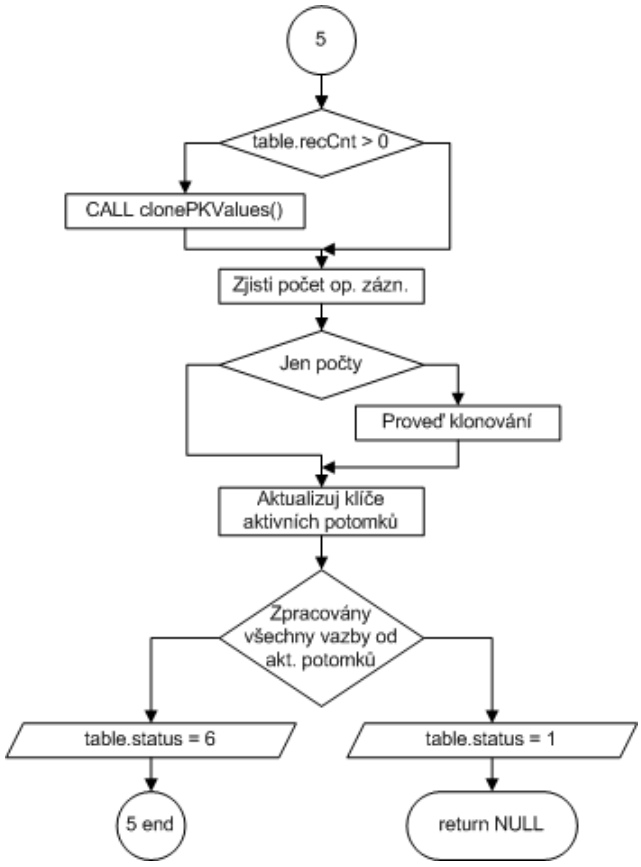
V opačném případě opět probíhá detekce kandidátů, kteří se nemohou stát operačními záznamy. Pro nerekurzivní tabulky a pro rekurzivní tabulky bez pokročilého zpracování jsou při detekci příslušní kandidáti ihned odstraněni. Pro rekurzivní tabulky s pokročilým zpracováním se příslušní kandidáti hned nemažou; pokud jsou nějací detekováni, je vyvolána pomocná metoda pokročilého zpracování rekurzivních tabulek.

Detekce je nutná z toho důvodu, že kandidáti rodičovské tabulky, vložené některou z aktivních vazeb, nemuseli splnit podmínky pro operační záznam, tudíž nebyli naklonováni a nelze na ně z tabulky *table* odkazovat - hodnota příslušného cizího klíče v tabulce kandidátů bude NULL. Pokud je vazba povinná, je toto důvodem k vyřazení kandidáta.

Nakonec tabulka přejde do stavu 5.

Zpracování stavu 5

Ve stavu 5 je prováděno zjišťování počtu operačních záznamů a případně i jejich klonování.



Obrázek 3.9

Pokud tabulka nemá rekurzivní vazby, je volána metoda na naplnění nových hodnot primárních klíčů. V opačném případě již tyto hodnoty byly naplněny při zpracování rekurzivních vazeb.

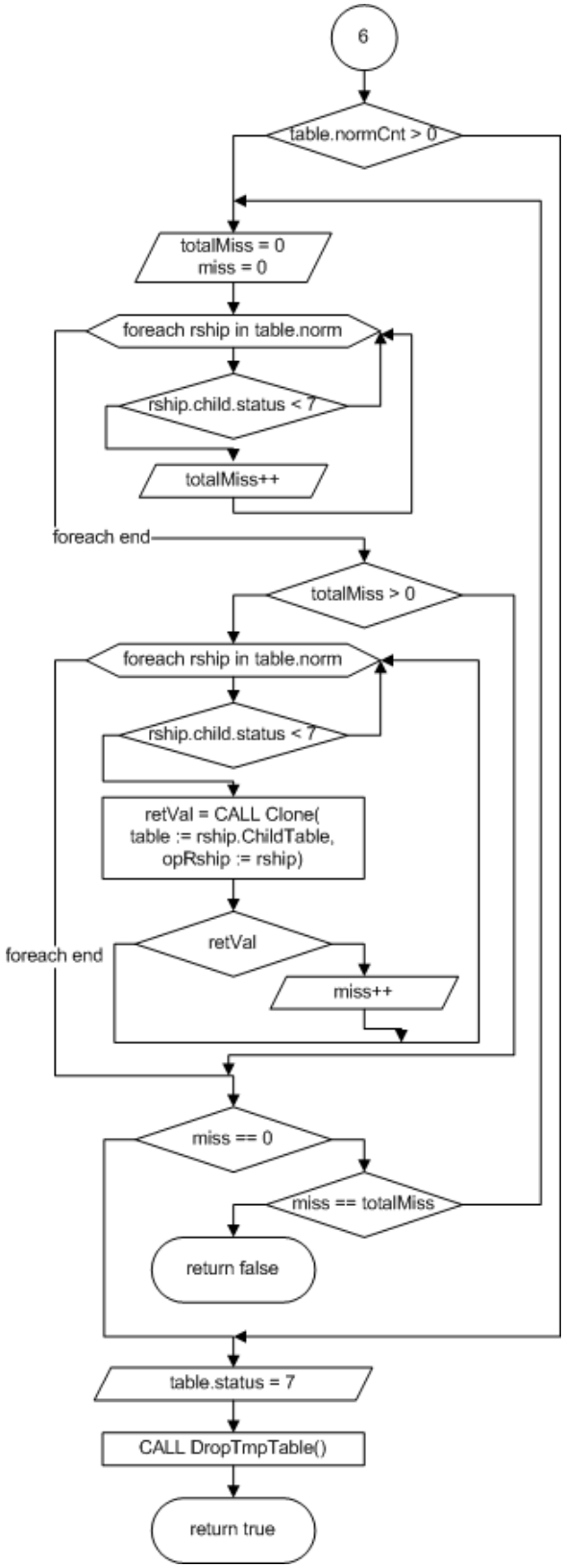
Následně je zjištěn počet operačních záznamů a v případě, že operace nebyla spuštěna právě pouze za účelem zjištění počtů, je provedeno i vlastní klonování.

Poté jsou aktualizovány hodnoty cizích klíčů kandidátů dětských tabulek, které mají na tabulku *table* aktivní vazbu.

Nakonec v případě, že byly zpracovány všechny aktivní vazby od potomků, přejde tabulka do stavu 6. Jinak přejde do stavu 1 a metoda je ukončena s návratovou hodnotou NULL.

Zpracování stavu 6

Ve stavu 6 jsou zpracovávány běžné vazby k potomkům tabulky *table*.


6
table.normCnt > 0
totalMiss = 0
miss = 0
foreach rship in table.norm
rship.child.status < 7
totalMiss++
foreach end
totalMiss > 0
foreach rship in table.norm
rship.child.status < 7
retVal = CALL Clone(
table := rship.ChildTable,
opRship := rship)
retVal
foreach end
miss++
miss == 0
miss == totalMiss
return false
table.status = 7
CALL DropTmpTable()
return true

Obrázek 3.10

Zpracování probíhá v cyklu, kde na jeho začátku je zjištěno, kolik dětských tabulek není zpracováno. Následně je pro každou nezpracovanou dětskou tabulku rekurzivně volána metoda *Clone*. Pokud vrátí false, tzn. pokud nedojde k úplnému zpracování dětské tabulky, je zvýšen čítač nezpracovaných tabulek. Nedojde-li ke zpracování ani jedné z tabulek, je metoda ukončena s návratovou hodnotou false. Některá z vyšších úrovní rekurze však způsobí zpracování dalších tabulek, takže při opětovném volání již může být tabulka úspěšně zpracována. Dříve či později tak dojde ke zpracování celé operační oblasti.

Po zpracování všech dětských tabulek přejde tabulka *table* do stavu 7 – zpracování dokončeno – a je odstraněna dočasná tabulka. Metoda končí s návratovou hodnotou true. Pokud je *table* hlavní tabulkou, je operace klonování dokončena.

3.5 Algoritmus mazání dat

Mazání je realizováno metodou

```
bool? Delete(table, opRship)
```

Parametr *table* je aktuálně zpracovávaná operační tabulka, parametr *opRship* je vazba, přes kterou bylo zpracování spuštěno. Při prvním volání metody v rámci celé operace je vazba NULL, tabulka je nastavena na hlavní tabulku.

3.5.1 Řízení operace

Algoritmus ke každé tabulce eviduje stav. Tabulka v průběhu zpracování přejde postupně 7 různými stavy:

0. nezpracovaná
1. zpracování běžných vazeb k rodičům a aktivních vazeb od potomků
2. zpracování rekurzivních vazeb
3. zpracování běžných vazeb k potomkům
4. vlastní vykonání operace
5. zpracování aktivních vazeb k rodičům
6. zpracovaná

Dále je třeba evidovat stav k běžným vazbám:

0. nezpracovaná
1. zpracovaná

3.5.2 Obsah dočasné tabulky

Dočasná tabulka pro operaci mazání obsahuje

- sloupce primárního klíče

- pro každou běžnou a rekurzivní vazbu sloupec *[název vazby]* pro uložení příznaku, zda je operační kandidát kandidátem dané vazby (hodnota 1)
- při pokročilém zpracování rekurzivních vazeb sloupec *#[název vazby]* uchovávající příznak, zda je kandidát první úrovní (hodnota 1)
- sloupec *_active* pro uchování příznaku, zda byl kandidát, případně jeho rodič či aktivní potomek, vložen do operace aktivní vazbou (hodnota 1)
- sloupec *_delete* pro uchování příznaku, zda má být operační kandidát odstraněn z tabulky kandidátů, tj. kandidát se nemůže stát operačním záznamem (hodnota 1)
- sloupec *_deletable* pro uchování příznaku, zda je operační záznam odstranitelný (hodnota 1)

3.5.3 Význam proměnných a vlastností v diagramech

V diagramech se vyskytují některé proměnné a vlastnosti tabulek a vazeb. Zde je jejich přehled

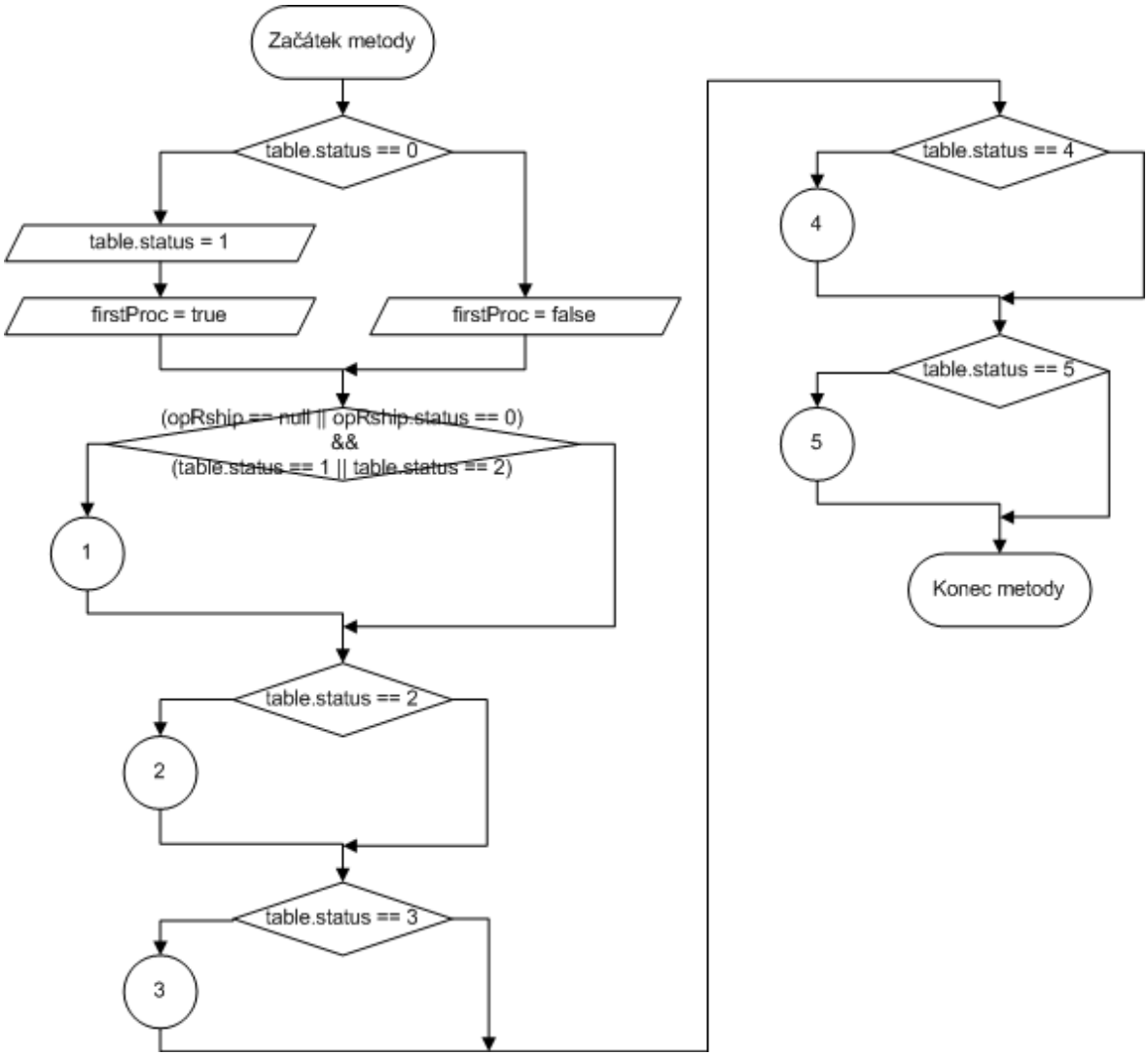
table.status	status tabulky
table.norm	běžné vazby k rodičům
table.normCnt	počet běžných vazeb k rodičům
table.normProc	počet zpracovaných běžných vazeb k rodičům
table.pass	aktivní vazby od potomků
table.passCnt	počet aktivních vazeb od potomků
table.passProc	počet zpracovaných aktivních vazeb od potomků
table.act	aktivní vazby k rodičům
table.actCnt	počet aktivních vazeb k rodičům
table.rec	rekurzivní vazby
table.recCnt	počet rekurzivních vazeb
table.isMain	příznak, zda jde o hlavní tabulku
table.preserve1st	příznak, zda budou zpracovány 1. úrovně rekurzivních vazeb
opRship.status	status vazby
opRship.isRecursive	příznak, zda jde o rekurzivní vazbu
opRship.isActive	příznak, zda jde o aktivní vazbu
opRship.activeProc	počet vyvolaných zpracování aktivní vazby
opRship.activeProcCnt	počet teoretického maximálního počtu volání zpracování aktivní vazby
firstProc	příznak, zda jde o první zpracování tabulky (tj. tabulka ještě nebyla zpracována žádnou operační vazbou)
activeCand	příznak, zda je metoda volána za účelem detekce operačních kandidátů aktivní vazby k rodiči před smazáním záznamů v dětské tabulce

Tabulka 3.2

3.5.4 Popis algoritmu

Poznámka: levá větev rozhodovacího bloku v diagramech značí splnění podmínky, pravá pak nesplnění podmínky.

Kostra metody Delete

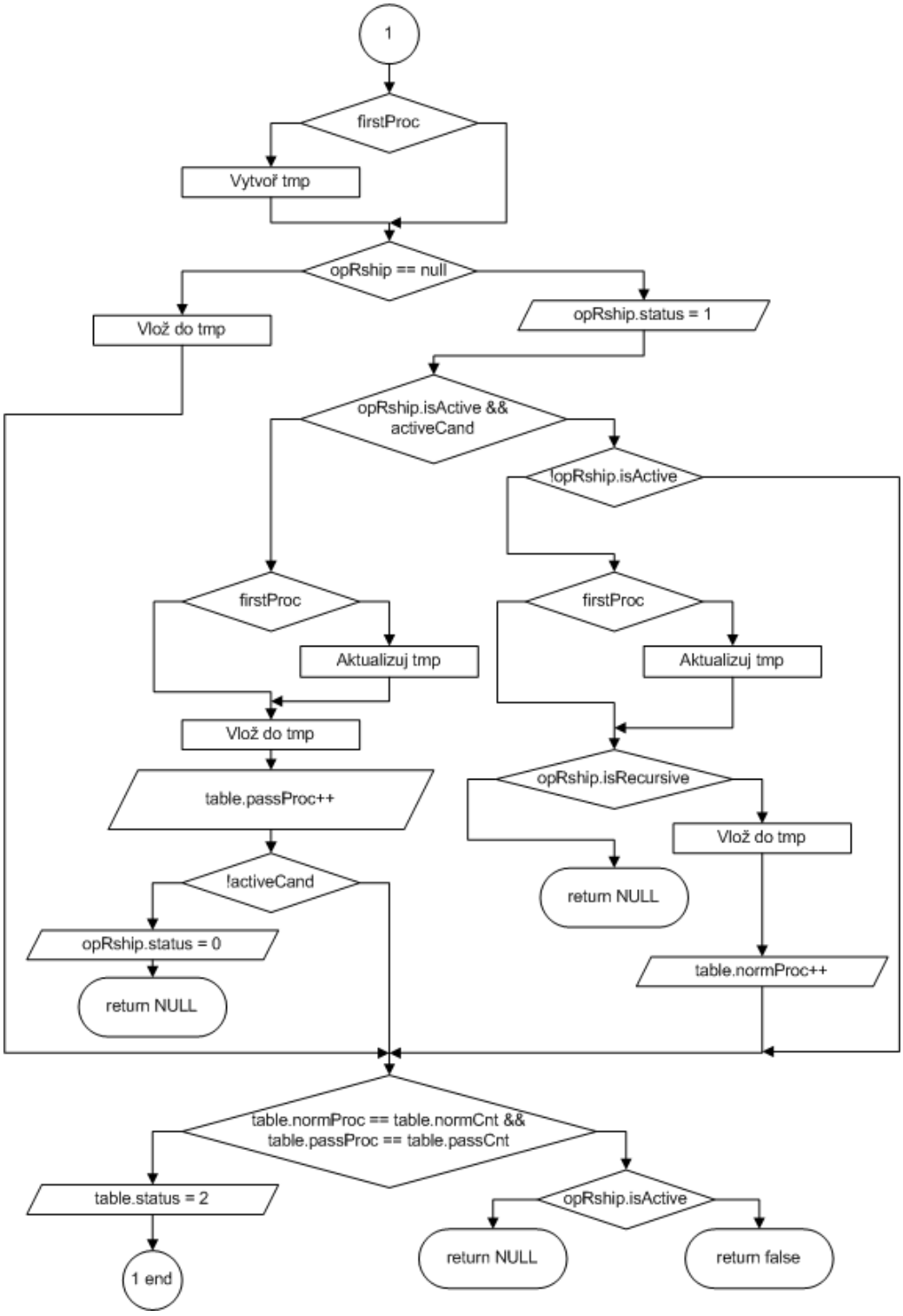


Obrázek 3.11

Zpracování stavu 1

Pro tabulku ve stavu 1 jsou detekováni operační kandidáti jednotlivých vazeb. Tato část metody *Delete* je rovněž využívána pro zpracování rekurzivních vazeb u tabulek ve stavu 2.

Zpracování podobné zpracování stavu 1 v operaci klonování.



Obrázek 3.12

Jedná-li se o první zpracování tabulky *table*, je vytvořena dočasná tabulka operačních kandidátů. Následuje samotná detekce kandidátů.

Pokud je *opRship* NULL, jde o první zpracování hlavní tabulky a stačí tak provést pouze vložení operačních kandidátů, tj. hodnot primárních klíčů záznamů tabulky *table* vyhovujících příslušným omezujícím podmínkám.

Pokud je vazba *opRship* aktivní a nejedná se o první zpracování tabulky, již existujícím kandidátům je nastaven příznak *_active* na hodnotu 1. Následuje samotné vložení kandidátů aktivní vazbou, tj. vložení hodnot příslušných primárních klíčů. Příznak *_active* je u těchto kandidátů nastaven na hodnotu 1. Je aktualizován čítač zpracovaných aktivních vazeb od potomků. Pokud bylo zpracování aktivní vazby vyvoláno těsně před smazáním záznamů v dětské tabulce, je metoda ukončena s návratovou hodnotou NULL – řízení je předáno vyšší úrovni rekurze, tj. zpracování oné dětské tabulky.

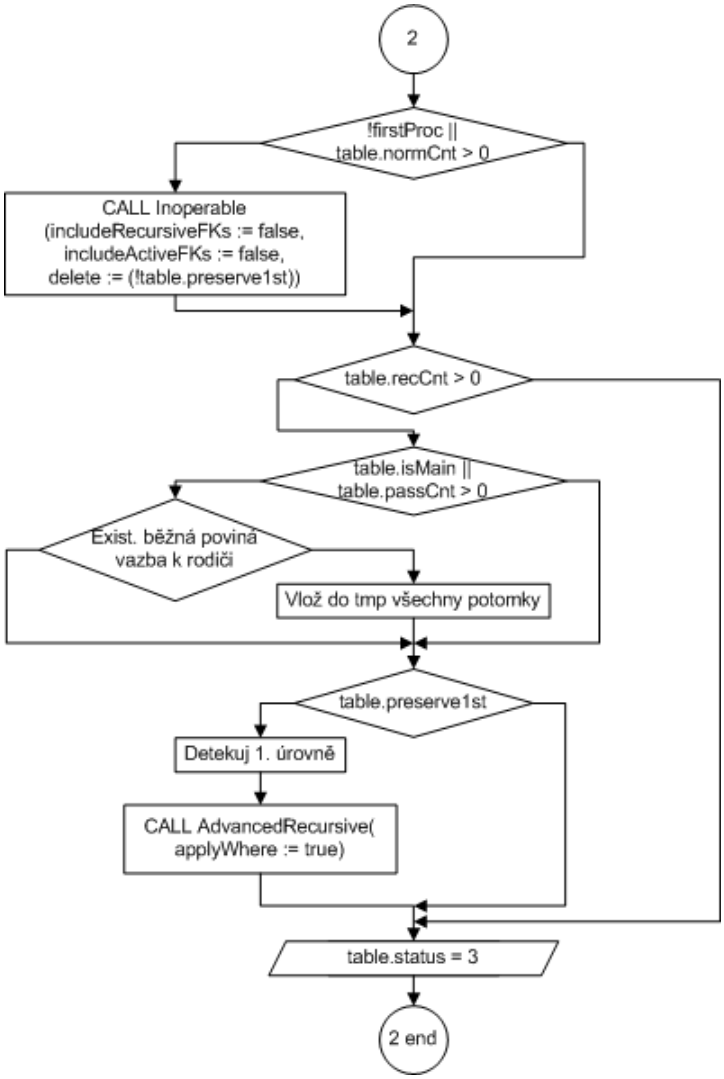
V případě, že *opRship* není aktivní a nejedná se o první zpracování tabulky, již existujícím kandidátům je nastaven příznak ve sloupci příslušné vazby na hodnotu 1. Další zpracování závisí na tom, je-li vazba rekurzivní. Pokud ano, je metoda ukončena s návratovou hodnotu NULL. Naopak u nerekurzivní vazby jsou vloženi kandidáty této vazby, tj. primární klíče příslušných záznamů tabulky *table*. Sloupec příslušné vazby těchto kandidátů je nastaven na hodnotu 1. Čítač zpracovaných běžných vazeb od rodičů je zvýšen.

Po ukončení samotné detekce závisí další postup na míře zpracování tabulky. Jsou-li zpracovány všechny běžné operační vazby k rodičům a všechny aktivní vazby od potomků, tabulka přejde do stavu 2. V opačném případě je metoda ukončena s návratovou hodnotou NULL pro aktivní vazbu, respektive false pro běžnou vazbu.

Zpracování stavu 2

Tabulka ve stavu 2 zpracovává své rekurzivní vazby.

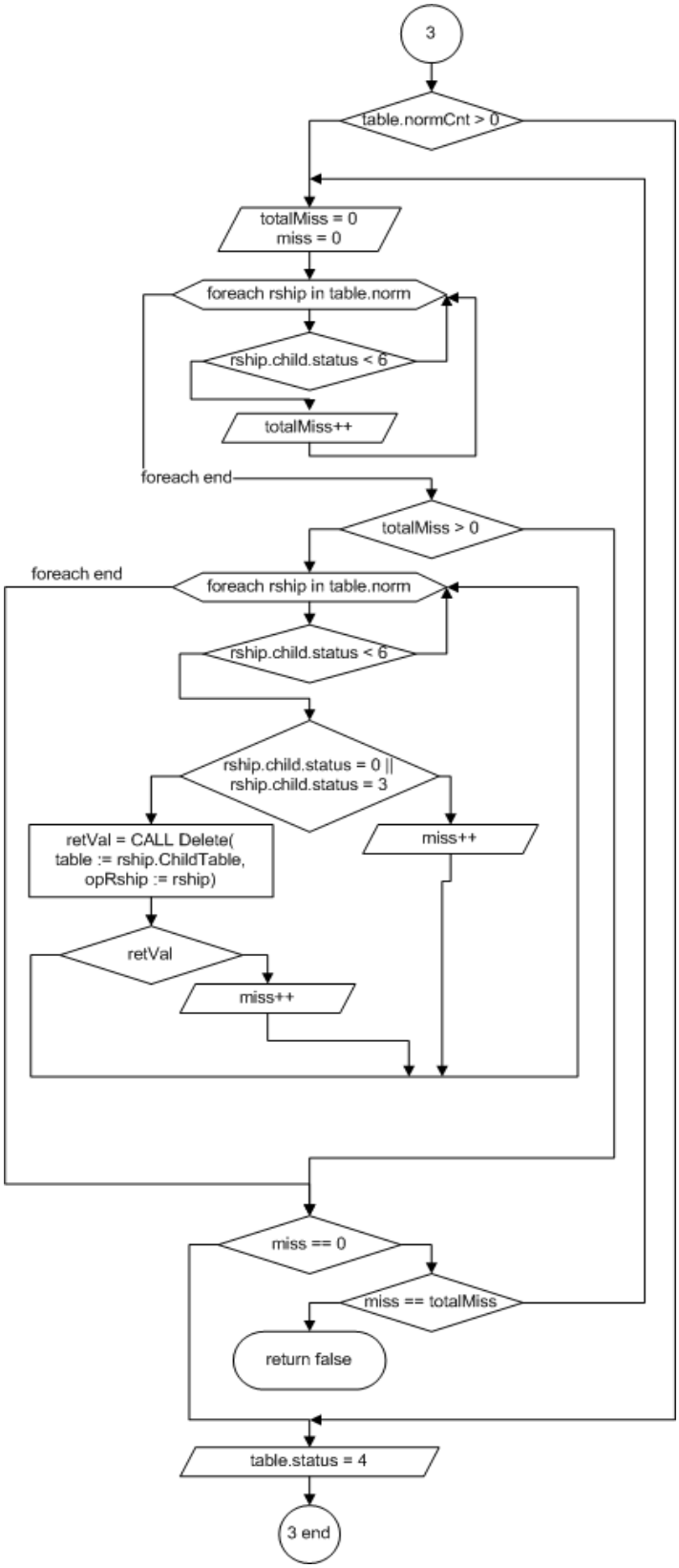
Zpracování je stejné jako u operace klonování, vyjmuto je pouze nastavení hodnot klonovaných primárních klíčů.



Obrázek 3.13

Zpracování stavu 3

Ve stavu 3 jsou zpracovávány běžné vazby k potomkům.


3
table.normCnt > 0
totalMiss = 0
miss = 0
foreach rship in table.norm
rship.child.status < 6
totalMiss++
foreach end
totalMiss > 0
foreach end
foreach rship in table.norm
rship.child.status < 6
rship.child.status = 0 ||
rship.child.status = 3
retVal = CALL Delete(
table := rship.ChildTable,
opRship := rship)
miss++
retVal
miss++
miss == 0
miss == totalMiss
return false
table.status = 4
3 end

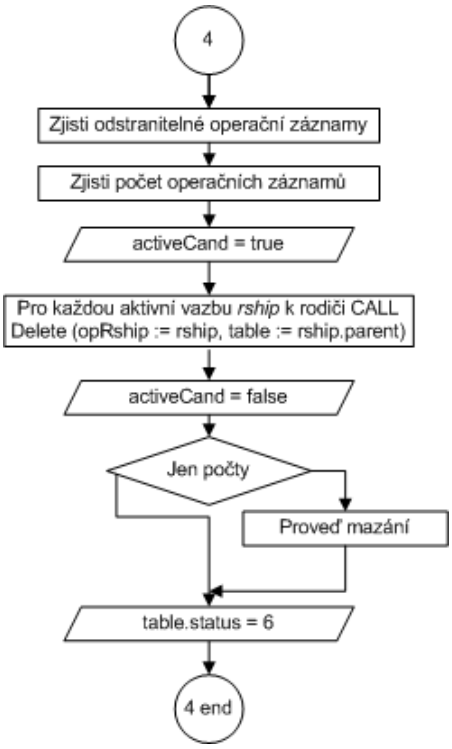
Obrázek 3.14

Zpracování probíhá v cyklu, kde na jeho začátku je zjištěno, kolik dětských tabulek není zpracováno. Následně je pro každou nezpracovanou dětskou tabulku rekurzivně volána metoda *Delete*. Pokud vrátí false, tzn. pokud nedojde k úplnému zpracování dětské tabulky, je zvýšen čítač nezpracovaných tabulek. Nedojde-li ke zpracování ani jedné z tabulek, je metoda ukončena s návratovou hodnotou false. Některá z vyšších úrovní rekurze však způsobí zpracování dalších tabulek, takže při opětovném volání již může být tabulka úspěšně zpracována. Dříve či později tak dojde ke zpracování celé operační oblasti.

Po zpracování všech dětských tabulek přejde tabulka do stavu 4.

Zpracování stavu 4

Ve stavu 4 je zjištěn počet operačních kandidátů a případně je provedeno vlastní smazání dat.



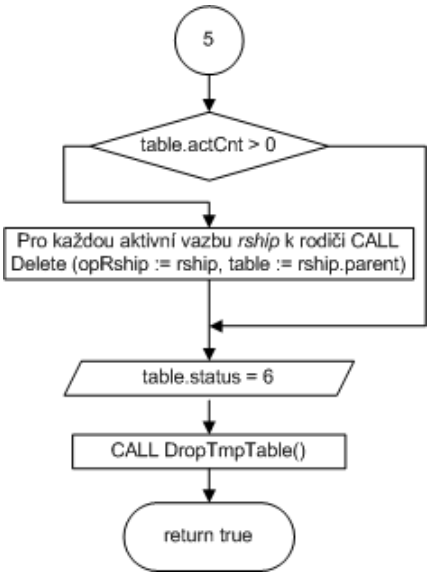
Obrázek 3.15

Nejprve proběhne detekce odstranitelných operačních záznamů. Následuje zjištění počtu operačních záznamů a odstranitelných operačních záznamů. Poté je volána metoda *Delete* nad aktivními vazbami k rodičům za účelem detekce operačních kandidátů v rodičovské tabulce, protože tyto po odstranění operačních záznamů z tabulky *table* nebude možné.

Pokud operace není spuštěna pouze pro zjištění počtu záznamů, proběhne vlastní odstranění dat a tabulka přejde do stavu 5.

Zpracování stavu 5

Ve stavu 5 jsou zpracovávány vazby k aktivním rodičům.



Obrázek 3.16

Po zpracování všech vazeb je zavoláno odstranění dočasné tabulky a metoda končí s návratovou hodnotou true. Je-li *table* hlavní tabulkou, je operace dokončena.

4 Nástroj DbsTools

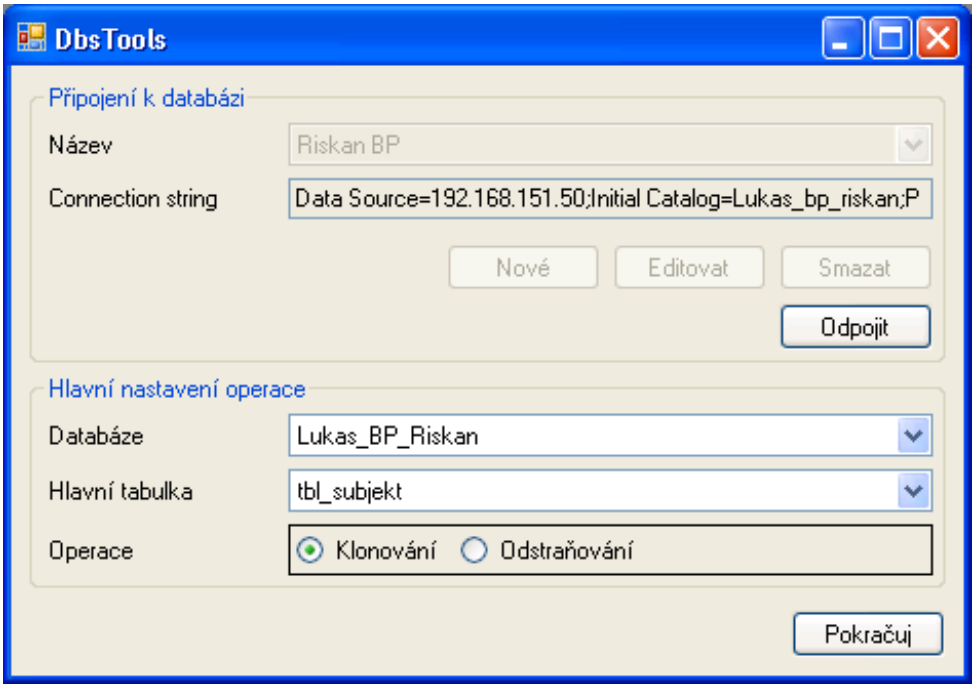
Samotný nástroj s názvem *DbsTools* vznikala paralelně s vývojem operačních algoritmů. Je vytvořena v jazyce C# nad platformou .NET 3.5. K používání není nutná instalace.

V této kapitole uvedu stručný manuál k ovládání aplikace.

4.1 Připojení a volba operace

Po spuštění programu je zobrazen dialog pro připojení k databázi. Můžeme vybrat některý z existujících pojmenovaných *connection stringů* (připojovacích řetězců), editovat, přidávat a odstraňovat je. Pojmenovaná připojení jsou uložena v souboru connections.xml.

Po připojení vybereme hlavní tabulku a požadovanou operaci. Stiskem tlačítka *Pokračuj* přejdeme do fáze konfigurace vybrané operace.



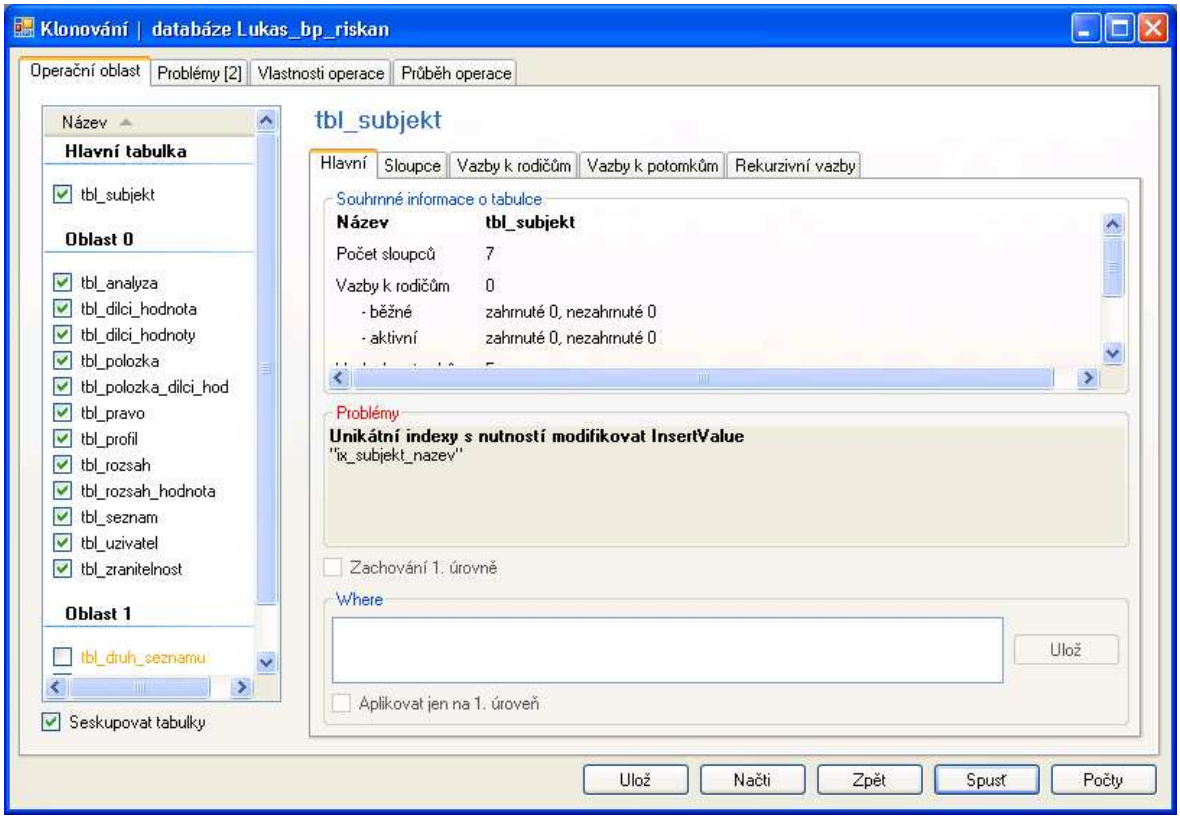
Obrázek 4.1

4.2 Konfigurace a ovládání operace

Okno pro konfiguraci a ovládání operace je rozděleno do čtyř záložek

1. Operační oblast – konfigurace operační oblasti

2. Problémy – seznam problémů bránících spuštění operace
3. Vlastnosti operace – konfigurace vlastností na úrovni celé operace
4. Průběh operace – sledování průběhu spuštěné operace, zobrazení posledního výsledku



Obrázek 4.2

4.2.1 Operační oblast

Okno je rozděleno na dvě části, v levé vidíme seznam tabulek, v pravé části pak pět záložek sloužících k zobrazení informací o vybrané tabulce a k její konfiguraci.

Seznam tabulek

Oranžové tabulky (bude použito i jako označení v dalším textu) patří do detekované oblasti, neoranžové odškrtnuté (explicitně vyjmuté) tabulky a tabulky s šedým pozadím (implicitně vyjmuté) patří do plně detekované oblasti, ostatní tabulky patří do operační oblasti.

Dvojklikem na řádek seznamu nebo klikem na zaškrtávací pole řádku je možné oranžovou tabulkou po potvrzení příslušných dialogů rozšířit detekovanou, případně i operační oblasti (pokud je tabulka zpracovatelná).

Dvojklikem na řádek seznamu nebo klikem na zaškrtávací pole řádku je možné neoranžovou tabulku po potvrzení příslušného dialogu vložit do operační oblasti, respektive vyjmout z operační oblasti.

Záložka „Hlavní“

Záložka obsahuje souhrnné informace o tabulce. Jsou zde v případě potřeby také zobrazovány problémy týkající se vybrané tabulky.

Pro rekurzivní tabulku lze nastavit vlastnost *Zachování 1. úrovně*.

Na záložce můžeme dále nastavit omezující podmínku *Where*. U rekurzivních tabulek můžeme zvolit, zda jí budeme aplikovat na všechny úrovně, nebo pouze na první úroveň. Pro aplikaci na první úroveň u klonování musí být vybrána možnost *Zachování 1. úrovně*.

Záložka „Sloupce“

Největší část zaujímá přehled sloupců tabulky. *PK* udává, zda je sloupec součástí primárního klíče tabulky. *FK* udává, zda je sloupec součástí cizího klíče. Pokud ano, je zobrazen název klíče. Pokud se navíc jedná o klíč operační vazby, je název zobrazen tučně. *UIX* udává, zda je sloupec součástí alespoň jednoho unikátního indexu (vyjma primárního klíče). Seznam unikátních indexů pro vybraný sloupec se nachází ve spodní části okna.

Při klonování lze vybranému sloupci (pokud není cizím klíčem a není identity) nastavit vkládanou hodnotu *Insert Value*. U textových datových typů můžeme zadat také speciální hodnotu <<*index*>>, která indexuje hodnotu klonovaného záznamu (např. originální záznam má ve sloupci název hodnotu „Software“, klon bude mít hodnotu „Software 1“). V případě, že indexové číslo spolu s původní hodnotou nevleze do délky pole, je původní hodnota zprava krácená o potřebný počet znaků. Pokud ani toto nepomůže, operace selže.

Záložky „Vazby k rodičům“ a „Vazby k potomkům“

Záložky slouží k zobrazení a konfiguraci příslušných vazeb tabulky.

Oranžové vazby patří do detekované oblasti, nezaškrtnuté vazby a vazby s šedým pozadím patří do plně detekované oblasti, ostatní vazby pak patří do operační oblasti. Kurzívou jsou znázorněny aktivní vazby.

Dvojklikem na řádek nebo klikem na zaškrtávací pole řádku lze tabulkou na druhé straně oranžové vazby po potvrzení příslušných dialogů rozšířit plně detekovanou oblast, při splnění potřebných podmínek rovnou i operační oblast

Dvojklikem na řádek nebo klikem na zaškrtávací pole řádku lze vazbu vyjmout z operační oblasti, respektive vložit do operační oblasti.

Tlačítkem *Aktivní* lze u jedné nebo více neoranžových vazeb změnit typ vazby z běžné na aktivní a naopak.

Tlačítkem *Zpracování* lze jednu nebo více neoranžových vazeb přesunout z operační oblasti do plně detekované oblasti a naopak.

Tlačítkem *Rozšíření* lze jednou nebo více oranžovými vazbami rozšířit plně detekovanou, případně i operační oblast.

Záložka „Rekurzivní vazby“

Slouží pouze k zobrazení rekurzivních vazeb tabulky. Těmto vazbám nelze měnit žádné nastavení ani je z operační oblasti vyjmout.

4.2.2 Problémy

Záložka Problémy informuje o všech problémech, které brání provedení operace. U obou operací se jedná o cyklus mezi tabulkami, u operace klonování pak navíc

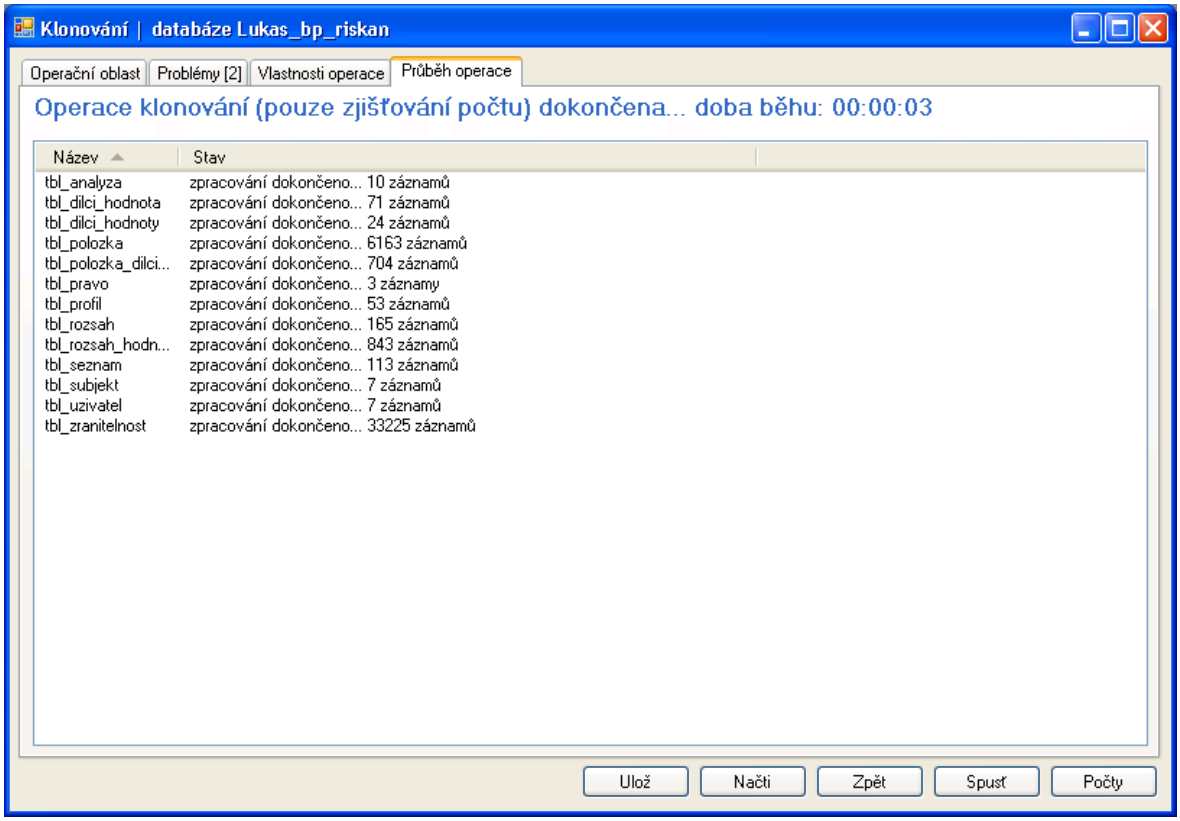
- Nutnost modifikace vkládané hodnoty alespoň jednoho sloupce primárního klíče
- Nutnost modifikace vkládané hodnoty alespoň jedeno sloupce unikátního indexu
- Nutnost rozšíření operační oblasti z důvodu unikátního indexu pouze na neklonovaných cizích klíčích
- Přítomnost identity sloupce v cizím klíči – tento problém nemá v nástroji řešení

4.2.3 Vlastnosti operace

Záložka slouží k nastavení operace jako celku. Můžeme nastavit vypnutí indexů na operačních tabulkách při manipulaci s daty (nedoporučuji) a timeout databázových příkazů (doporučuji nechat na 0).

4.2.4 Průběh operace

Záložka slouží ke sledování průběhu běžící operace a zobrazení výsledků posledního spuštění (operaci lze spouštět opakovaně – užitečné například při změně operační oblasti či omezujících podmínek).



Obrázek 4.3

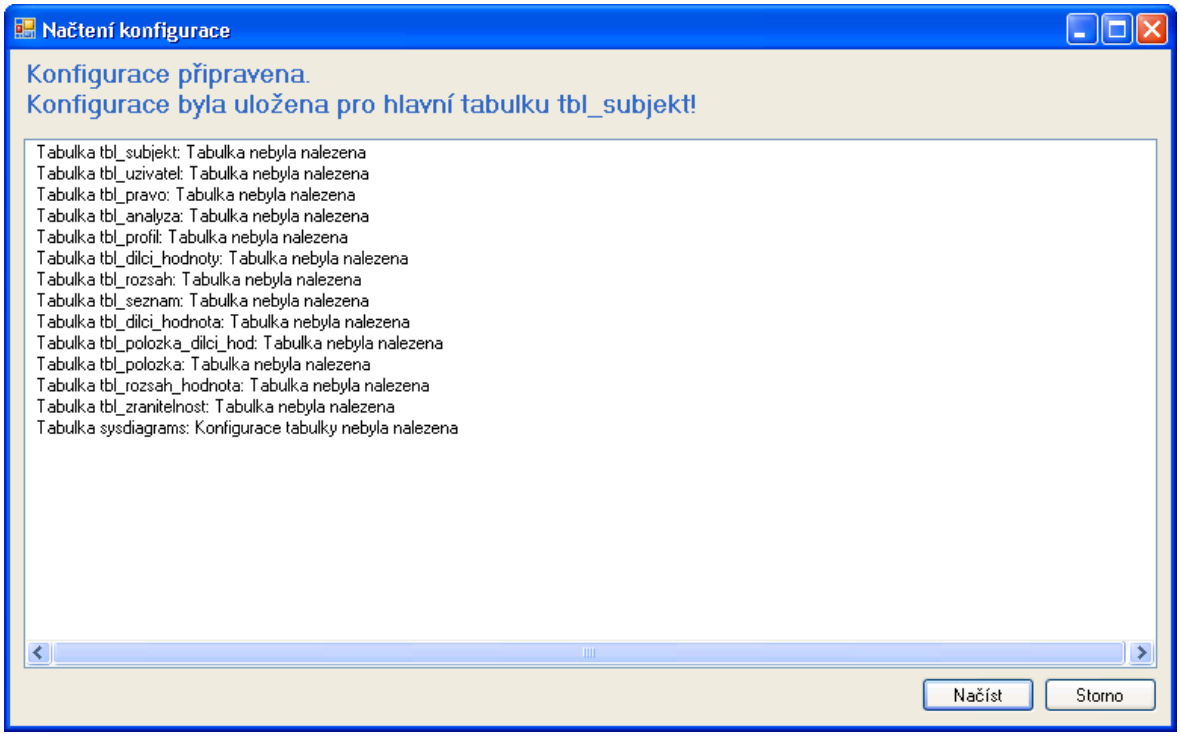
4.3 Uložení a načtení konfigurace

Jedním s požadavků na nástroj je podpora ukládání a načítání konfigurací.

Konfigurace jsou ukládány do souboru XML stiskem tlačítka *Ulož* ve spodní části okna.

K načtení pak slouží tlačítko *Načti*.

Ukládáno je veškeré nastavení operační oblasti včetně informací o tabulkách, vazbách a sloupcích v operační oblasti. Tyto informace pak při načítání slouží k detekci rozdílů mezi operační oblastí v konfiguračním souboru a operační oblastí načtenou v aplikaci. Detekované rozdíly jsou vypsány do okna, uživatel se na základě nich může sám rozhodnout, zda konfiguraci načte nebo ne.



Obrázek 4.4

4.4 Spuštění operace

Operaci spustíme tlačítkem *Spusť*. Operaci lze spustit až po vyřešení všech problémů.

Tlačítkem *Počty* spustíme operaci v režimu zjišťování počtu operačních záznamů.

Pozor: zjišťování počtu je oproti normálnímu spuštění operace rychlejší pouze o čas, který zabere manipulaci s daty v operačních tabulkách. Detekce operačních záznamů využívá stejný algoritmus jako běžné spuštění operace, pouze vynechá vlastní kopii či odstranění dat.

Tlačítkem *Zpět* se můžeme vrátit na úvodní „připojovací“ dialog.

4.5 Log

Operace vytvářejí log použitých SQL dotazů. Ten je umístěn v adresáři s aplikací a je pojmenován jako [operace]_[americké datum a čas].txt. Pod každým dotazem je uvedena doba běhu a počet ovlivněných záznamů.

Na konci logu je souhrnná informace o počtu zpracovaných záznamů.

5 Závěr

Předmětem bakalářské práce bylo vyvinutí algoritmů pro klonování a mazání dat a následné vytvoření desktopového nástroje pro konfiguraci a spouštění těchto operací.

Výsledkem je aplikace umožňující poměrně snadné klonování a mazání dat s dosti širokými možnostmi nastavení. Je možné upravovat operační oblast, stanovovat omezující podmínky pro libovolnou tabulku operační oblasti, definovat modifikační funkce pro klonované hodnoty. Nástroj detekuje problémy, které znemožňují úspěšné provedení operace. Konkrétní nastavení operace lze uložit do souboru XML, nastavení pro klonování a mazání jsou vzájemně kompatibilní. Načíst lze i konfiguraci uloženou pro jinou operační oblast – uživatel je informován o zjištěných rozdílech a může se rozhodnout, zda konfiguraci použije či nikoliv.

Praktické výsledky použití nástroje jsou zatím k dispozici pouze z vývoje a testování. To zatím neodhalilo žádné závažné nedostatky v podobě chybného klonování či mazání dat, zdá se tedy, že aktuální verzi již je možné (s příslušnou mírou opatrnosti) využívat v reálném provozu.

Podmínkou bylo, aby byl nástroj využitelný i pro větší objemy dat. Zde je problém v relativnosti slova „použitelný“. Vždy bude záležet na konkrétní situaci a nakonec i na konkrétním vývojáři, zda mu bude připadat čekání dlouhé např. několik desítek vteřin moc nebo málo. Například největším testovaným objemem klonovaných dat bylo přibližně 1,7 milionu záznamů v operační obsahující 44 tabulek, z nich 26 obsahujících data. Objem dat v jedné tabulce kolísal od jednotek záznamů až po statisíce. Výsledný čas na vývojovém serveru (Pentium 4 530, 4 GB RAM, Windows 2003 Server Standard Edition) k provedení celé operace byl něco přes 7 minut. Na první pohled je to poměrně dlouhá doba, na straně druhé – když vezmeme v úvahu množství indexů a strukturu databáze – je výsledný čas dobrý. Určitě zde existuje prostor pro ladění algoritmů, hlavně co se indexů na dočasných tabulkách kandidátů týče. K tomu je však třeba mít dostatek podkladů z praktického využití nástroje.

V současné verzi je pak třeba dořešit transakční zpracování. Zamezit změně dat v průběhu operace, obzvláště od momentu detekce operačních kandidátů, je pro správné fungování nutné. Momentálně je na začátku operace spuštěna transakce a je aplikován exkluzivní zámek na celou operační oblast. Tím může být velká část databáze vyřazena z provozu pro ostatní uživatele a to i na poměrně dlouhou dobu. Na druhou stranu se jedná o vývojový nástroj, jehož primárním účelem není využití v provozním prostředí informačního systému. Této problematice bude ještě věnována příslušná péče.

Pokud jde o možnosti dalšího rozvoje, tak již nyní mám k dispozici podněty od kolegů k úpravám převážně uživatelského rozhraní i k zajímavým úpravám samotných možností obou operací. Výhledově dále do budoucna je pak možné rozšířit nástroj o další operace, například generování testovacích dat, anonymizaci dat, generování SQL skriptů a podobně. Základem těchto operací by se mohl stát některý z právě vyvinutých operačních algoritmů. Je počítáno také s rozšířením na platformu Oracle. Zdrojové kódy jsou na toto z velké části připraveny.

Při vývoji jsem vycházel z mých dosavadních zkušeností s relačními databázemi. Vývoj šel postupně od jednoduchého klonování bez omezujících podmínek přes jejich podporu, pokročilé zpracování rekurzivních vazeb až k aktivním vazbám. Některé na první pohled snadné věci se změnily pomalu v „noční můru“ a naopak. Dost práce dala také implementace úprav operační oblasti nebo detekce potenciálních problémů.

Doufám, že se mi povedlo čtenářům alespoň trochu srozumitelně přiblížit problematiku, kterou jsem se zabýval.

Práce neobsahuje žádné citace z literatury ani z internetu. Vše uvedené pochází z mé hlavy, internet byl využíván jako jakási „technická podpora“ při řešení převážně programátorských problémů („Co umí třída ABC“, „Jak se používá třída DEF“ apod.) a při zjišťování informací o Microsoft SQL (zde posloužily výborně stránky http://msdn.microsoft.com).